

VoyagerOS64  
0.0.5 (Diagnostics Branch)

Generated by Doxygen 1.9.1



---

<b>1 Data Structure Index</b>	<b>1</b>
1.1 Data Structures . . . . .	1
<b>2 File Index</b>	<b>3</b>
2.1 File List . . . . .	3
<b>3 Data Structure Documentation</b>	<b>5</b>
3.1 AtomicLock Class Reference . . . . .	5
3.1.1 Detailed Description . . . . .	5
3.1.2 Constructor & Destructor Documentation . . . . .	5
3.1.2.1 AtomicLock() . . . . .	5
3.1.3 Member Function Documentation . . . . .	5
3.1.3.1 ForceLock() . . . . .	6
3.1.3.2 IsLocked() . . . . .	6
3.1.3.3 Lock() . . . . .	6
3.1.3.4 Unlock() . . . . .	6
3.2 cpu_local Struct Reference . . . . .	7
3.2.1 Detailed Description . . . . .	7
3.2.2 Field Documentation . . . . .	7
3.2.2.1 active . . . . .	7
3.2.2.2 bsp . . . . .	8
3.2.2.3 cpu_number . . . . .	8
3.2.2.4 idle_thread . . . . .	8
3.2.2.5 last_run_queue_index . . . . .	8
3.2.2.6 timer_function . . . . .	8
3.2.2.7 tss . . . . .	8
3.3 limine_5_level_paging_request Struct Reference . . . . .	9
3.3.1 Detailed Description . . . . .	9
3.3.2 Member Function Documentation . . . . .	9
3.3.2.1 LIMINE_PTR() . . . . .	9
3.3.3 Field Documentation . . . . .	9
3.3.3.1 id . . . . .	9
3.3.3.2 revision . . . . .	9
3.4 limine_boot_time_request Struct Reference . . . . .	10
3.4.1 Detailed Description . . . . .	10
3.4.2 Member Function Documentation . . . . .	10
3.4.2.1 LIMINE_PTR() . . . . .	10
3.4.3 Field Documentation . . . . .	10
3.4.3.1 id . . . . .	10
3.4.3.2 revision . . . . .	10
3.5 limine_bootloader_info_request Struct Reference . . . . .	11
3.5.1 Detailed Description . . . . .	11
3.5.2 Member Function Documentation . . . . .	11

---

3.5.2.1 LIMINE_PTR() . . . . .	11
3.5.3 Field Documentation . . . . .	11
3.5.3.1 id . . . . .	11
3.5.3.2 revision . . . . .	11
3.6 limine_bootloader_info_response Struct Reference . . . . .	12
3.6.1 Detailed Description . . . . .	12
3.6.2 Member Function Documentation . . . . .	12
3.6.2.1 LIMINE_PTR() [1/2] . . . . .	12
3.6.2.2 LIMINE_PTR() [2/2] . . . . .	12
3.6.3 Field Documentation . . . . .	12
3.6.3.1 revision . . . . .	12
3.7 limine_dtb_request Struct Reference . . . . .	13
3.7.1 Detailed Description . . . . .	13
3.7.2 Member Function Documentation . . . . .	13
3.7.2.1 LIMINE_PTR() . . . . .	13
3.7.3 Field Documentation . . . . .	13
3.7.3.1 id . . . . .	13
3.7.3.2 revision . . . . .	13
3.8 limine_dtb_response Struct Reference . . . . .	14
3.8.1 Detailed Description . . . . .	14
3.8.2 Member Function Documentation . . . . .	14
3.8.2.1 LIMINE_PTR() . . . . .	14
3.8.3 Field Documentation . . . . .	14
3.8.3.1 revision . . . . .	14
3.9 limine_efi_system_table_request Struct Reference . . . . .	14
3.9.1 Detailed Description . . . . .	15
3.9.2 Member Function Documentation . . . . .	15
3.9.2.1 LIMINE_PTR() . . . . .	15
3.9.3 Field Documentation . . . . .	15
3.9.3.1 id . . . . .	15
3.9.3.2 revision . . . . .	15
3.10 limine_efi_system_table_response Struct Reference . . . . .	16
3.10.1 Detailed Description . . . . .	16
3.10.2 Member Function Documentation . . . . .	16
3.10.2.1 LIMINE_PTR() . . . . .	16
3.10.3 Field Documentation . . . . .	16
3.10.3.1 revision . . . . .	16
3.11 limine_entry_point_request Struct Reference . . . . .	16
3.11.1 Detailed Description . . . . .	17
3.11.2 Member Function Documentation . . . . .	17
3.11.2.1 LIMINE_PTR() [1/2] . . . . .	17
3.11.2.2 LIMINE_PTR() [2/2] . . . . .	17

---

3.11.3 Field Documentation . . . . .	17
3.11.3.1 id . . . . .	17
3.11.3.2 revision . . . . .	18
3.12 limine_file Struct Reference . . . . .	18
3.12.1 Detailed Description . . . . .	19
3.12.2 Member Function Documentation . . . . .	19
3.12.2.1 LIMINE_PTR() [1/3] . . . . .	19
3.12.2.2 LIMINE_PTR() [2/3] . . . . .	19
3.12.2.3 LIMINE_PTR() [3/3] . . . . .	19
3.12.3 Field Documentation . . . . .	19
3.12.3.1 gpt_disk_uuid . . . . .	19
3.12.3.2 gpt_part_uuid . . . . .	19
3.12.3.3 mbr_disk_id . . . . .	20
3.12.3.4 media_type . . . . .	20
3.12.3.5 part_uuid . . . . .	20
3.12.3.6 partition_index . . . . .	20
3.12.3.7 revision . . . . .	20
3.12.3.8 size . . . . .	20
3.12.3.9 tftp_ip . . . . .	21
3.12.3.10 tftp_port . . . . .	21
3.12.3.11 unused . . . . .	21
3.13 limine_framebuffer Struct Reference . . . . .	21
3.13.1 Detailed Description . . . . .	22
3.13.2 Member Function Documentation . . . . .	22
3.13.2.1 LIMINE_PTR() [1/2] . . . . .	22
3.13.2.2 LIMINE_PTR() [2/2] . . . . .	22
3.13.3 Field Documentation . . . . .	22
3.13.3.1 blue_mask_shift . . . . .	22
3.13.3.2 blue_mask_size . . . . .	22
3.13.3.3 bpp . . . . .	22
3.13.3.4 edid_size . . . . .	23
3.13.3.5 green_mask_shift . . . . .	23
3.13.3.6 green_mask_size . . . . .	23
3.13.3.7 height . . . . .	23
3.13.3.8 memory_model . . . . .	23
3.13.3.9 pitch . . . . .	23
3.13.3.10 red_mask_shift . . . . .	24
3.13.3.11 red_mask_size . . . . .	24
3.13.3.12 unused . . . . .	24
3.13.3.13 width . . . . .	24
3.14 limine_framebuffer_request Struct Reference . . . . .	24
3.14.1 Detailed Description . . . . .	25

3.14.2 Member Function Documentation . . . . .	25
3.14.2.1 LIMINE_PTR() . . . . .	25
3.14.3 Field Documentation . . . . .	25
3.14.3.1 id . . . . .	25
3.14.3.2 revision . . . . .	25
3.15 limine_framebuffer_response Struct Reference . . . . .	25
3.15.1 Detailed Description . . . . .	26
3.15.2 Member Function Documentation . . . . .	26
3.15.2.1 LIMINE_PTR() . . . . .	26
3.15.3 Field Documentation . . . . .	26
3.15.3.1 framebuffer_count . . . . .	26
3.15.3.2 revision . . . . .	26
3.16 limine_hhd़m_request Struct Reference . . . . .	26
3.16.1 Detailed Description . . . . .	27
3.16.2 Member Function Documentation . . . . .	27
3.16.2.1 LIMINE_PTR() . . . . .	27
3.16.3 Field Documentation . . . . .	27
3.16.3.1 id . . . . .	27
3.16.3.2 revision . . . . .	27
3.17 limine_kernel_address_request Struct Reference . . . . .	28
3.17.1 Detailed Description . . . . .	28
3.17.2 Member Function Documentation . . . . .	28
3.17.2.1 LIMINE_PTR() . . . . .	28
3.17.3 Field Documentation . . . . .	28
3.17.3.1 id . . . . .	28
3.17.3.2 revision . . . . .	28
3.18 limine_kernel_file_request Struct Reference . . . . .	29
3.18.1 Detailed Description . . . . .	29
3.18.2 Member Function Documentation . . . . .	29
3.18.2.1 LIMINE_PTR() . . . . .	29
3.18.3 Field Documentation . . . . .	29
3.18.3.1 id . . . . .	29
3.18.3.2 revision . . . . .	29
3.19 limine_kernel_file_response Struct Reference . . . . .	30
3.19.1 Detailed Description . . . . .	30
3.19.2 Member Function Documentation . . . . .	30
3.19.2.1 LIMINE_PTR() . . . . .	30
3.19.3 Field Documentation . . . . .	30
3.19.3.1 revision . . . . .	30
3.20 limine_memmap_request Struct Reference . . . . .	30
3.20.1 Detailed Description . . . . .	31
3.20.2 Member Function Documentation . . . . .	31

---

3.20.2.1 LIMINE_PTR() . . . . .	31
3.20.3 Field Documentation . . . . .	31
3.20.3.1 id . . . . .	31
3.20.3.2 revision . . . . .	31
3.21 limine_memmap_response Struct Reference . . . . .	32
3.21.1 Detailed Description . . . . .	32
3.21.2 Member Function Documentation . . . . .	32
3.21.2.1 LIMINE_PTR() . . . . .	32
3.21.3 Field Documentation . . . . .	32
3.21.3.1 entry_count . . . . .	32
3.21.3.2 revision . . . . .	32
3.22 limine_module_request Struct Reference . . . . .	33
3.22.1 Detailed Description . . . . .	33
3.22.2 Member Function Documentation . . . . .	33
3.22.2.1 LIMINE_PTR() . . . . .	33
3.22.3 Field Documentation . . . . .	33
3.22.3.1 id . . . . .	33
3.22.3.2 revision . . . . .	33
3.23 limine_module_response Struct Reference . . . . .	34
3.23.1 Detailed Description . . . . .	34
3.23.2 Member Function Documentation . . . . .	34
3.23.2.1 LIMINE_PTR() . . . . .	34
3.23.3 Field Documentation . . . . .	34
3.23.3.1 module_count . . . . .	34
3.23.3.2 revision . . . . .	34
3.24 limine_rsdp_request Struct Reference . . . . .	35
3.24.1 Detailed Description . . . . .	35
3.24.2 Member Function Documentation . . . . .	35
3.24.2.1 LIMINE_PTR() . . . . .	35
3.24.3 Field Documentation . . . . .	35
3.24.3.1 id . . . . .	35
3.24.3.2 revision . . . . .	35
3.25 limine_rsdp_response Struct Reference . . . . .	36
3.25.1 Detailed Description . . . . .	36
3.25.2 Member Function Documentation . . . . .	36
3.25.2.1 LIMINE_PTR() . . . . .	36
3.25.3 Field Documentation . . . . .	36
3.25.3.1 revision . . . . .	36
3.26 limine_smbios_request Struct Reference . . . . .	36
3.26.1 Detailed Description . . . . .	37
3.26.2 Member Function Documentation . . . . .	37
3.26.2.1 LIMINE_PTR() . . . . .	37

3.26.3 Field Documentation . . . . .	37
3.26.3.1 id . . . . .	37
3.26.3.2 revision . . . . .	37
3.27 limine_smbios_response Struct Reference . . . . .	38
3.27.1 Detailed Description . . . . .	38
3.27.2 Member Function Documentation . . . . .	38
3.27.2.1 LIMINE_PTR() [1/2] . . . . .	38
3.27.2.2 LIMINE_PTR() [2/2] . . . . .	38
3.27.3 Field Documentation . . . . .	38
3.27.3.1 revision . . . . .	38
3.28 limine_smp_request Struct Reference . . . . .	39
3.28.1 Detailed Description . . . . .	39
3.28.2 Member Function Documentation . . . . .	39
3.28.2.1 LIMINE_PTR() . . . . .	39
3.28.3 Field Documentation . . . . .	39
3.28.3.1 flags . . . . .	39
3.28.3.2 id . . . . .	39
3.28.3.3 revision . . . . .	40
3.29 limine_stack_size_request Struct Reference . . . . .	40
3.29.1 Detailed Description . . . . .	40
3.29.2 Member Function Documentation . . . . .	40
3.29.2.1 LIMINE_PTR() . . . . .	40
3.29.3 Field Documentation . . . . .	40
3.29.3.1 id . . . . .	41
3.29.3.2 revision . . . . .	41
3.29.3.3 stack_size . . . . .	41
3.30 limine_terminal Struct Reference . . . . .	41
3.30.1 Detailed Description . . . . .	41
3.30.2 Member Function Documentation . . . . .	42
3.30.2.1 LIMINE_PTR() . . . . .	42
3.30.3 Field Documentation . . . . .	42
3.30.3.1 columns . . . . .	42
3.30.3.2 rows . . . . .	42
3.31 limine_terminal_request Struct Reference . . . . .	42
3.31.1 Detailed Description . . . . .	43
3.31.2 Member Function Documentation . . . . .	43
3.31.2.1 LIMINE_PTR() [1/2] . . . . .	43
3.31.2.2 LIMINE_PTR() [2/2] . . . . .	43
3.31.3 Field Documentation . . . . .	43
3.31.3.1 id . . . . .	43
3.31.3.2 revision . . . . .	43
3.32 limine_terminal_response Struct Reference . . . . .	43

---

3.32.1 Detailed Description . . . . .	44
3.32.2 Member Function Documentation . . . . .	44
3.32.2.1 LIMINE_PTR() [1/2] . . . . .	44
3.32.2.2 LIMINE_PTR() [2/2] . . . . .	44
3.32.3 Field Documentation . . . . .	44
3.32.3.1 revision . . . . .	44
3.32.3.2 terminal_count . . . . .	45
3.33 out_fct_wrap_type Struct Reference . . . . .	45
3.33.1 Detailed Description . . . . .	45
3.33.2 Field Documentation . . . . .	45
3.33.2.1 arg . . . . .	45
3.33.2.2 fct . . . . .	45
3.34 ScopedLock Class Reference . . . . .	46
3.34.1 Detailed Description . . . . .	46
3.34.2 Constructor & Destructor Documentation . . . . .	46
3.34.2.1 ScopedLock() . . . . .	46
3.34.2.2 ~ScopedLock() . . . . .	47
3.35 term_context Struct Reference . . . . .	47
3.35.1 Detailed Description . . . . .	48
3.35.2 Field Documentation . . . . .	48
3.35.2.1 autoflush . . . . .	48
3.35.2.2 bold . . . . .	49
3.35.2.3 callback . . . . .	49
3.35.2.4 charsets . . . . .	49
3.35.2.5 clear . . . . .	49
3.35.2.6 cols . . . . .	49
3.35.2.7 control_sequence . . . . .	49
3.35.2.8 csi . . . . .	50
3.35.2.9 current_charset . . . . .	50
3.35.2.10 current_primary . . . . .	50
3.35.2.11 dec_private . . . . .	50
3.35.2.12 deinit . . . . .	50
3.35.2.13 disable_cursor . . . . .	50
3.35.2.14 discard_next . . . . .	51
3.35.2.15 double_buffer_flush . . . . .	51
3.35.2.16 enable_cursor . . . . .	51
3.35.2.17 esc_values . . . . .	51
3.35.2.18 esc_values_i . . . . .	51
3.35.2.19 escape . . . . .	51
3.35.2.20 escape_offset . . . . .	52
3.35.2.21 full_refresh . . . . .	52
3.35.2.22 g_select . . . . .	52

3.35.2.23 get_cursor_pos . . . . .	52
3.35.2.24 in_bootloader . . . . .	52
3.35.2.25 insert_mode . . . . .	52
3.35.2.26 move_character . . . . .	53
3.35.2.27 raw_putchar . . . . .	53
3.35.2.28 restore_state . . . . .	53
3.35.2.29 reverse_video . . . . .	53
3.35.2.30 revscroll . . . . .	53
3.35.2.31 rows . . . . .	53
3.35.2.32 rrr . . . . .	54
3.35.2.33 save_state . . . . .	54
3.35.2.34 saved_cursor_x . . . . .	54
3.35.2.35 saved_cursor_y . . . . .	54
3.35.2.36 saved_state_bold . . . . .	54
3.35.2.37 saved_state_current_charset . . . . .	54
3.35.2.38 saved_state_current_primary . . . . .	55
3.35.2.39 saved_state_reverse_video . . . . .	55
3.35.2.40 scroll . . . . .	55
3.35.2.41 scroll_bottom_margin . . . . .	55
3.35.2.42 scroll_enabled . . . . .	55
3.35.2.43 scroll_top_margin . . . . .	55
3.35.2.44 set_cursor_pos . . . . .	56
3.35.2.45 set_text_bg . . . . .	56
3.35.2.46 set_text_bg_bright . . . . .	56
3.35.2.47 set_text_bg_default . . . . .	56
3.35.2.48 set_text_bg_rgb . . . . .	56
3.35.2.49 set_text_fg . . . . .	56
3.35.2.50 set_text_fg_bright . . . . .	57
3.35.2.51 set_text_fg_default . . . . .	57
3.35.2.52 set_text_fg_rgb . . . . .	57
3.35.2.53 swap_palette . . . . .	57
3.35.2.54 tab_size . . . . .	57
<b>4 File Documentation</b> . . . . .	<b>59</b>
<b>4.1 src/alloc.cpp File Reference</b> . . . . .	<b>59</b>
<b>4.1.1 Function Documentation</b> . . . . .	60
<b>4.1.1.1 liballoc_alloc()</b> . . . . .	60
<b>4.1.1.2 liballoc_free()</b> . . . . .	60
<b>4.1.1.3 liballoc_lock()</b> . . . . .	61
<b>4.1.1.4 liballoc_unlock()</b> . . . . .	62
<b>4.1.2 Variable Documentation</b> . . . . .	62
<b>4.1.2.1 liballocLock</b> . . . . .	62

---

4.2 alloc.cpp . . . . .	63
4.3 src/cpudet-clean.c File Reference . . . . .	63
4.3.1 Macro Definition Documentation . . . . .	64
4.3.1.1 cpuid . . . . .	64
4.3.2 Function Documentation . . . . .	65
4.3.2.1 detect_cpu() . . . . .	65
4.3.2.2 do_amd() . . . . .	65
4.3.2.3 do_intel() . . . . .	66
4.3.2.4 printregs() . . . . .	66
4.3.3 Variable Documentation . . . . .	67
4.3.3.1 Intel . . . . .	67
4.3.3.2 Intel_Other . . . . .	67
4.4 cpudet-clean.c . . . . .	68
4.5 src/cpuUtils.c File Reference . . . . .	72
4.5.1 Function Documentation . . . . .	74
4.5.1.1 check_acpi() . . . . .	74
4.5.1.2 check_apic() . . . . .	75
4.5.1.3 check_ds() . . . . .	76
4.5.1.4 check_fpu() . . . . .	77
4.5.1.5 check_htt() . . . . .	77
4.5.1.6 check_mca() . . . . .	78
4.5.1.7 check_mce() . . . . .	79
4.5.1.8 check_msр() . . . . .	80
4.5.1.9 check_pae() . . . . .	80
4.5.1.10 check_pcid() . . . . .	81
4.5.1.11 check_sse() . . . . .	82
4.5.1.12 check_tm() . . . . .	83
4.5.1.13 check_vmx() . . . . .	83
4.5.1.14 check_xsave() . . . . .	84
4.5.1.15 cpuid_check_acpi() . . . . .	85
4.5.1.16 cpuid_check_apic() . . . . .	85
4.5.1.17 cpuid_check_avx() . . . . .	85
4.5.1.18 cpuid_check_ds() . . . . .	86
4.5.1.19 cpuid_check_fpu() . . . . .	86
4.5.1.20 cpuid_check_fxsr() . . . . .	86
4.5.1.21 cpuid_check_htt() . . . . .	86
4.5.1.22 cpuid_check_mca() . . . . .	87
4.5.1.23 cpuid_check_mce() . . . . .	87
4.5.1.24 cpuid_check_msр() . . . . .	87
4.5.1.25 cpuid_check_oxsave() . . . . .	87
4.5.1.26 cpuid_check_pae() . . . . .	88
4.5.1.27 cpuid_check_pcid() . . . . .	88

---

4.5.1.28 cpuid_check_sse()	88
4.5.1.29 cpuid_check_tm()	89
4.5.1.30 cpuid_check_vmx()	89
4.5.1.31 cpuid_check_xsave()	89
4.5.1.32 cpuid_readout()	90
4.5.1.33 fpu_init()	91
4.5.1.34 get_model()	91
4.5.1.35 get_vendor()	92
4.5.1.36 halt()	92
4.5.1.37 no_sse()	92
4.5.1.38 no_xsave()	93
4.5.1.39 vendor_str1()	94
4.5.1.40 vendor_str2()	94
4.5.1.41 vendor_str3()	94
4.5.2 Variable Documentation	94
4.5.2.1 CPU_vendor	94
4.5.2.2 first_run	94
4.6 cpuUtils.c	95
4.7 src/drivers/keyboard/keyboard.c File Reference	99
4.7.1 Macro Definition Documentation	100
4.7.1.1 KBD_STACK_SIZE	100
4.7.2 Function Documentation	100
4.7.2.1 k_getchar()	100
4.7.2.2 kbd_pop()	101
4.7.2.3 kbd_push()	101
4.7.2.4 keyboard_handler()	102
4.7.2.5 keyboard_handler_2()	103
4.7.2.6 keyboard_init()	104
4.7.2.7 read_input()	105
4.7.3 Variable Documentation	106
4.7.3.1 k_char	106
4.7.3.2 kbd_stack	106
4.7.3.3 kbd_top	106
4.7.3.4 reader_lock	106
4.8 keyboard.c	107
4.9 src/frameallocator.c File Reference	109
4.9.1 Function Documentation	110
4.9.1.1 frame_free()	111
4.9.1.2 frame_free_multiple()	111
4.9.1.3 frame_lock()	112
4.9.1.4 frame_lock_multiple()	113
4.9.1.5 frame_request()	113

---

4.9.1.6 frame_request_multiple()	114
4.9.1.7 free_ram()	114
4.9.1.8 halt()	115
4.9.1.9 MemoryMapTypeString()	115
4.9.1.10 print_frame_bitmap()	115
4.9.1.11 read_memory_map()	116
4.9.1.12 reserved_ram()	117
4.9.1.13 used_ram()	117
4.9.2 Variable Documentation	117
4.9.2.1 bitmapSize	117
4.9.2.2 frameBitmap	118
4.9.2.3 freeMemory	118
4.9.2.4 initialized	118
4.9.2.5 reservedMemory	118
4.9.2.6 usedMemory	118
4.10 frameallocator.c	119
4.11 src/gdt.c File Reference	122
4.11.1 Function Documentation	123
4.11.1.1 breakpoint()	123
4.11.1.2 halt()	123
4.11.1.3 LoadGDT_Stage1()	123
4.11.1.4 serial_debug()	124
4.11.2 Variable Documentation	124
4.11.2.1 desc	125
4.11.2.2 gdt	125
4.11.2.3 ist1Stack	125
4.11.2.4 ist2Stack	125
4.11.2.5 rsp0	125
4.11.2.6 tss	125
4.11.2.7 TssStack	126
4.12 gdt.c	126
4.13 src/heap.c File Reference	128
4.13.1 Function Documentation	129
4.13.1.1 k_heapBMAddBlock()	129
4.13.1.2 k_heapBMAddBlockEx()	129
4.13.1.3 k_heapBMAalloc()	130
4.13.1.4 k_heapBMAallocBound()	130
4.13.1.5 k_heapBMFree()	131
4.13.1.6 k_heapBMGetBMSize()	131
4.13.1.7 k_heapBMGetNID()	132
4.13.1.8 k_heapBMInit()	132
4.13.1.9 k_heapBMSet()	132

---

4.14 heap.c . . . . .	132
4.15 src/idt.c File Reference . . . . .	135
4.15.1 Function Documentation . . . . .	136
4.15.1.1 idt_allocate_vector() . . . . .	136
4.15.1.2 idt_free_vector() . . . . .	137
4.15.1.3 idt_init() . . . . .	137
4.15.1.4 idt_set_descriptor() . . . . .	138
4.15.2 Variable Documentation . . . . .	138
4.15.2.1 idt . . . . .	138
4.15.2.2 idtr . . . . .	138
4.15.2.3 isr_stub_table . . . . .	138
4.15.2.4 vectors . . . . .	139
4.16 idt.c . . . . .	139
4.17 src/include/bitmap.h File Reference . . . . .	140
4.17.1 Function Documentation . . . . .	140
4.17.1.1 bitmap_get() . . . . .	141
4.17.1.2 bitmap_set() . . . . .	141
4.18 bitmap.h . . . . .	142
4.19 src/include/cpuUtils.h File Reference . . . . .	142
4.19.1 Data Structure Documentation . . . . .	144
4.19.1.1 struct cpu_ctxt . . . . .	144
4.19.2 Macro Definition Documentation . . . . .	145
4.19.2.1 _CPU_UTILS_H . . . . .	145
4.19.2.2 CPUID_VENDOR_AMD . . . . .	145
4.19.2.3 CPUID_VENDOR_AO486 . . . . .	145
4.19.2.4 CPUID_VENDOR_BHYVE . . . . .	145
4.19.2.5 CPUID_VENDOR_CENTAUR . . . . .	146
4.19.2.6 CPUID_VENDOR_CYRIX . . . . .	146
4.19.2.7 CPUID_VENDOR_ELBRUS . . . . .	146
4.19.2.8 CPUID_VENDOR_HYGON . . . . .	146
4.19.2.9 CPUID_VENDOR_HYPERV . . . . .	146
4.19.2.10 CPUID_VENDOR_INTEL . . . . .	146
4.19.2.11 CPUID_VENDOR_KVM . . . . .	147
4.19.2.12 CPUID_VENDOR_NEXGEN . . . . .	147
4.19.2.13 CPUID_VENDOR_NSC . . . . .	147
4.19.2.14 CPUID_VENDOR_OLDAMD . . . . .	147
4.19.2.15 CPUID_VENDOR_OLDAO486 . . . . .	147
4.19.2.16 CPUID_VENDOR_OLDTRANSMETA . . . . .	147
4.19.2.17 CPUID_VENDOR_PARALLELS . . . . .	148
4.19.2.18 CPUID_VENDOR_PARALLELS_ALT . . . . .	148
4.19.2.19 CPUID_VENDOR_QEMU . . . . .	148
4.19.2.20 CPUID_VENDOR_QNX . . . . .	148

---

4.19.2.21 CPUID_VENDOR_RISE . . . . .	148
4.19.2.22 CPUID_VENDOR_SIS . . . . .	148
4.19.2.23 CPUID_VENDOR_TRANSMETA . . . . .	149
4.19.2.24 CPUID_VENDOR_UMC . . . . .	149
4.19.2.25 CPUID_VENDOR_VIA . . . . .	149
4.19.2.26 CPUID_VENDOR_VIRTUALBOX . . . . .	149
4.19.2.27 CPUID_VENDOR_VMWARE . . . . .	149
4.19.2.28 CPUID_VENDOR_VORTEX . . . . .	149
4.19.2.29 CPUID_VENDOR_XEN . . . . .	150
4.19.2.30 CPUID_VENDOR_ZHAOXIN . . . . .	150
4.19.3 Function Documentation . . . . .	150
4.19.3.1 cpu_init() . . . . .	150
4.19.3.2 cpuid_readout() . . . . .	150
4.19.3.3 detect_cpu() . . . . .	151
4.19.3.4 fxrstor() . . . . .	152
4.19.3.5 fxsave() . . . . .	152
4.19.3.6 get_fs_base() . . . . .	153
4.19.3.7 get_gs_base() . . . . .	153
4.19.3.8 get_kernel_gs_base() . . . . .	153
4.19.3.9 get_model() . . . . .	154
4.19.3.10 interrupt_state() . . . . .	154
4.19.3.11 rdmsr() . . . . .	154
4.19.3.12 set_fs_base() . . . . .	155
4.19.3.13 set_gs_base() . . . . .	155
4.19.3.14 set_kernel_gs_base() . . . . .	156
4.19.3.15 this_cpu() . . . . .	156
4.19.3.16 wrmsr() . . . . .	157
4.19.3.17 xrstor() . . . . .	157
4.19.3.18 xsave() . . . . .	157
4.19.4 Variable Documentation . . . . .	157
4.19.4.1 CPU_vendor . . . . .	158
4.19.4.2 fpu_bank_size . . . . .	158
4.19.4.3 fpu_rest . . . . .	158
4.19.4.4 fpu_save . . . . .	158
4.19.4.5 sysenter . . . . .	158
4.20 cpuUtils.h . . . . .	158
4.21 src/include/drivers/keyboard/keyboard.h File Reference . . . . .	161
4.21.1 Macro Definition Documentation . . . . .	161
4.21.1.1 KEYBOARD_DATA_PORT . . . . .	161
4.21.1.2 KEYBOARD_STATUS_PORT . . . . .	162
4.21.2 Function Documentation . . . . .	162
4.21.2.1 k_getchar() . . . . .	162

---

4.21.2.2 <code>kbd_pop()</code>	162
4.21.2.3 <code>keyboard_handler()</code>	163
4.21.2.4 <code>keyboard_handler_2()</code>	163
4.21.2.5 <code>keyboard_init()</code>	165
4.21.2.6 <code>read_input()</code>	165
4.22 <code>keyboard.h</code>	167
4.23 <code>src/include/drivers/keyboard/keyboard_map.h</code> File Reference	167
4.23.1 Variable Documentation	168
4.23.1.1 <code>keyboard_charmap</code>	168
4.23.1.2 <code>keyboard_map</code>	168
4.24 <code>keyboard_map.h</code>	169
4.25 <code>src/include/gdt.h</code> File Reference	172
4.25.1 Data Structure Documentation	174
4.25.1.1 <code>struct GDT_Desc</code>	174
4.25.1.2 <code>struct GDT_Entry</code>	174
4.25.1.3 <code>struct GDT_Entry_16</code>	174
4.25.1.4 <code>struct GDT_Entry_32</code>	174
4.25.1.5 <code>struct TSS_Entry</code>	175
4.25.1.6 <code>struct GDT</code>	175
4.25.2 Macro Definition Documentation	176
4.25.2.1 <code>GDTAccess16Code</code>	176
4.25.2.2 <code>GDTAccess16Data</code>	176
4.25.2.3 <code>GDTAccess32Code</code>	176
4.25.2.4 <code>GDTAccess32Data</code>	176
4.25.2.5 <code>GDTAccessDPL</code>	177
4.25.2.6 <code>GDTAccessKernelCode</code>	177
4.25.2.7 <code>GDTAccessKernelData</code>	177
4.25.2.8 <code>GDTAccessUserCode</code>	177
4.25.2.9 <code>GDTAccessUserData</code>	177
4.25.2.10 <code>GDTKernelBaseSelector</code>	177
4.25.2.11 <code>GDTTSSSegment</code>	178
4.25.2.12 <code>GDTUserBaseSelector</code>	178
4.25.3 Enumeration Type Documentation	178
4.25.3.1 <code>GDTAccessFlag</code>	178
4.25.4 Function Documentation	178
4.25.4.1 <code>LoadGDT_Stage1()</code>	178
4.25.5 Variable Documentation	179
4.25.5.1 <code>rsp0</code>	179
4.26 <code>gdt.h</code>	179
4.27 <code>src/include/global_defs.h</code> File Reference	180
4.27.1 Macro Definition Documentation	181
4.27.1.1 <code>ALIGN_16BIT</code>	181

---

4.27.1.2 ALIGN_4K . . . . .	182
4.27.1.3 CAS . . . . .	182
4.27.1.4 PACKED . . . . .	182
4.27.2 Typedef Documentation . . . . .	182
4.27.2.1 mode_t . . . . .	182
4.28 global_defs.h . . . . .	182
4.29 src/include/heap.h File Reference . . . . .	183
4.29.1 Data Structure Documentation . . . . .	184
4.29.1.1 struct _KHEAPBLOCKBM . . . . .	184
4.29.1.2 struct _KHEAPBM . . . . .	184
4.29.2 Typedef Documentation . . . . .	185
4.29.2.1 KHEAPBLOCKBM . . . . .	185
4.29.2.2 KHEAPBM . . . . .	185
4.29.3 Function Documentation . . . . .	185
4.29.3.1 k_heapBMAddBlock() . . . . .	186
4.29.3.2 k_heapBMAddBlockEx() . . . . .	186
4.29.3.3 k_heapBMAalloc() . . . . .	187
4.29.3.4 k_heapBMAallocBound() . . . . .	187
4.29.3.5 k_heapBMFree() . . . . .	188
4.29.3.6 k_heapBMGetBMSize() . . . . .	188
4.29.3.7 k_heapBMInit() . . . . .	188
4.29.3.8 k_heapBMSet() . . . . .	188
4.30 heap.h . . . . .	189
4.31 src/include/idt.h File Reference . . . . .	189
4.31.1 Data Structure Documentation . . . . .	190
4.31.1.1 struct __attribute__ . . . . .	190
4.31.2 Macro Definition Documentation . . . . .	191
4.31.2.1 IDT_CPU_EXCEPTION_COUNT . . . . .	191
4.31.2.2 IDT_DESCRIPTOR_CALL . . . . .	191
4.31.2.3 IDT_DESCRIPTOR_EXCEPTION . . . . .	191
4.31.2.4 IDT_DESCRIPTOR_EXTERNAL . . . . .	191
4.31.2.5 IDT_DESCRIPTOR_PRESENT . . . . .	192
4.31.2.6 IDT_DESCRIPTOR_RING1 . . . . .	192
4.31.2.7 IDT_DESCRIPTOR_RING2 . . . . .	192
4.31.2.8 IDT_DESCRIPTOR_RING3 . . . . .	192
4.31.2.9 IDT_DESCRIPTOR_X16_INTERRUPT . . . . .	192
4.31.2.10 IDT_DESCRIPTOR_X16_TRAP . . . . .	192
4.31.2.11 IDT_DESCRIPTOR_X32_INTERRUPT . . . . .	193
4.31.2.12 IDT_DESCRIPTOR_X32_TASK . . . . .	193
4.31.2.13 IDT_DESCRIPTOR_X32_TRAP . . . . .	193
4.31.2.14 IDT_HDW_INTERRUPT_COUNT . . . . .	193
4.31.2.15 IDT_MAX_DESCRIPTORS . . . . .	193

---

4.31.3 Function Documentation . . . . .	193
4.31.3.1 idt_allocate_vector() . . . . .	193
4.31.3.2 idt_free_vector() . . . . .	194
4.31.3.3 idt_init() . . . . .	194
4.31.3.4 idt_reload() . . . . .	195
4.31.3.5 idt_set_descriptor() . . . . .	195
4.32 idt.h . . . . .	195
4.33 src/include/interrupts.h File Reference . . . . .	196
4.33.1 Data Structure Documentation . . . . .	197
4.33.1.1 struct isr_xframe_t . . . . .	197
4.33.1.2 struct isr_xframe_t.control_registers . . . . .	197
4.33.1.3 struct isr_xframe_t.general_registers . . . . .	198
4.33.1.4 struct isr_xframe_t.base_frame . . . . .	198
4.34 interrupts.h . . . . .	198
4.35 src/include/io.h File Reference . . . . .	199
4.35.1 Macro Definition Documentation . . . . .	199
4.35.1.1 _IO_H . . . . .	200
4.35.2 Function Documentation . . . . .	200
4.35.2.1 inb() . . . . .	200
4.35.2.2 io_wait() . . . . .	200
4.35.2.3 outb() . . . . .	201
4.36 io.h . . . . .	201
4.37 src/include/kernel.h File Reference . . . . .	202
4.37.1 Function Documentation . . . . .	203
4.37.1.1 clear_screen() . . . . .	203
4.37.1.2 print_prompt() . . . . .	204
4.37.2 Variable Documentation . . . . .	204
4.37.2.1 bootspace . . . . .	204
4.37.2.2 early_term . . . . .	205
4.37.2.3 kerror_mode . . . . .	205
4.38 kernel.h . . . . .	205
4.39 src/include/KernelUtils.h File Reference . . . . .	205
4.39.1 Data Structure Documentation . . . . .	206
4.39.1.1 struct kswitches . . . . .	206
4.39.2 Macro Definition Documentation . . . . .	207
4.39.2.1 _KERNEL_UTILS_H . . . . .	207
4.39.3 Function Documentation . . . . .	207
4.39.3.1 get_memory_size() . . . . .	207
4.39.3.2 init_memory() . . . . .	207
4.39.3.3 print_memmap() . . . . .	208
4.39.3.4 print_memory() . . . . .	209
4.39.3.5 system_readout() . . . . .	210

---

4.39.4 Variable Documentation . . . . .	210
4.39.4.1 fbr_req . . . . .	210
4.39.4.2 k_mode . . . . .	210
4.39.4.3 memmap_req . . . . .	211
4.39.4.4 term_context . . . . .	211
4.40 KernelUtils.h . . . . .	211
4.41 src/include/lib/hashmap.h File Reference . . . . .	212
4.41.1 Macro Definition Documentation . . . . .	213
4.41.1.1 _HASHMAP_H . . . . .	213
4.41.1.2 HASHMAP_DELETE . . . . .	213
4.41.1.3 HASHMAP_GET . . . . .	214
4.41.1.4 HASHMAP_INIT . . . . .	214
4.41.1.5 HASHMAP_INSERT . . . . .	214
4.41.1.6 HASHMAP_KEY_DATA_MAX . . . . .	214
4.41.1.7 HASHMAP_REMOVE . . . . .	214
4.41.1.8 HASHMAP_SGET . . . . .	215
4.41.1.9 HASHMAP_SINSERT . . . . .	215
4.41.1.10 HASHMAP_SREMOVE . . . . .	215
4.41.1.11 HASHMAP_TYPE . . . . .	216
4.41.2 Function Documentation . . . . .	216
4.41.2.1 hash() . . . . .	216
4.42 hashmap.h . . . . .	216
4.43 src/include/lib/vector.h File Reference . . . . .	220
4.43.1 Macro Definition Documentation . . . . .	221
4.43.1.1 VECTOR_ENSURE_LENGTH . . . . .	221
4.43.1.2 VECTOR_FIND . . . . .	222
4.43.1.3 VECTOR_FOR_EACH . . . . .	222
4.43.1.4 VECTOR_INIT . . . . .	223
4.43.1.5 VECTOR_INSERT . . . . .	223
4.43.1.6 VECTOR_INVALID_INDEX . . . . .	224
4.43.1.7 VECTOR_ITEM . . . . .	224
4.43.1.8 VECTOR_PUSH_BACK . . . . .	224
4.43.1.9 VECTOR_PUSH_FRONT . . . . .	224
4.43.1.10 VECTOR_REMOVE . . . . .	225
4.43.1.11 VECTOR_REMOVE_BY_VALUE . . . . .	225
4.43.1.12 VECTOR_TYPE . . . . .	225
4.44 vector.h . . . . .	226
4.45 src/include/liballoc.h File Reference . . . . .	228
4.45.1 Data Structure Documentation . . . . .	229
4.45.1.1 struct boundary_tag . . . . .	229
4.45.2 Macro Definition Documentation . . . . .	229
4.45.2.1 NULL . . . . .	229

---

4.45.3 Function Documentation . . . . .	229
4.45.3.1 <code>calloc()</code> . . . . .	230
4.45.3.2 <code>free()</code> . . . . .	230
4.45.3.3 <code>liballoc_alloc()</code> . . . . .	231
4.45.3.4 <code>liballoc_free()</code> . . . . .	231
4.45.3.5 <code>liballoc_lock()</code> . . . . .	232
4.45.3.6 <code>liballoc_unlock()</code> . . . . .	232
4.45.3.7 <code>malloc()</code> . . . . .	233
4.45.3.8 <code>realloc()</code> . . . . .	234
4.46 <code>liballoc.h</code> . . . . .	235
4.47 <code>src/include/limine.h</code> File Reference . . . . .	235
4.47.1 Data Structure Documentation . . . . .	238
4.47.1.1 <code>struct limine_uuid</code> . . . . .	238
4.47.1.2 <code>struct limine_stack_size_response</code> . . . . .	238
4.47.1.3 <code>struct limine_hhdm_response</code> . . . . .	238
4.47.1.4 <code>struct limine_5_level_paging_response</code> . . . . .	239
4.47.1.5 <code>struct limine_memmap_entry</code> . . . . .	239
4.47.1.6 <code>struct limine_entry_point_response</code> . . . . .	239
4.47.1.7 <code>struct limine_boot_time_response</code> . . . . .	239
4.47.1.8 <code>struct limine_kernel_address_response</code> . . . . .	239
4.47.2 Macro Definition Documentation . . . . .	240
4.47.2.1 <code>LIMINE_5_LEVEL_PAGING_REQUEST</code> . . . . .	240
4.47.2.2 <code>LIMINE_BOOT_TIME_REQUEST</code> . . . . .	240
4.47.2.3 <code>LIMINE_BOOTLOADER_INFO_REQUEST</code> . . . . .	240
4.47.2.4 <code>LIMINE_COMMON_MAGIC</code> . . . . .	240
4.47.2.5 <code>LIMINE_DTB_REQUEST</code> . . . . .	240
4.47.2.6 <code>LIMINE_EFI_SYSTEM_TABLE_REQUEST</code> . . . . .	241
4.47.2.7 <code>LIMINE_ENTRY_POINT_REQUEST</code> . . . . .	241
4.47.2.8 <code>LIMINE_FRAMEBUFFER_REQUEST</code> . . . . .	241
4.47.2.9 <code>LIMINE_FRAMEBUFFER_RGB</code> . . . . .	241
4.47.2.10 <code>LIMINE_HHDM_REQUEST</code> . . . . .	241
4.47.2.11 <code>LIMINE_KERNEL_ADDRESS_REQUEST</code> . . . . .	241
4.47.2.12 <code>LIMINE_KERNEL_FILE_REQUEST</code> . . . . .	242
4.47.2.13 <code>LIMINE_MEDIA_TYPE_GENERIC</code> . . . . .	242
4.47.2.14 <code>LIMINE_MEDIA_TYPE_OPTICAL</code> . . . . .	242
4.47.2.15 <code>LIMINE_MEDIA_TYPE_TFTP</code> . . . . .	242
4.47.2.16 <code>LIMINE_MEMMAP_ACPI_NVS</code> . . . . .	242
4.47.2.17 <code>LIMINE_MEMMAP_ACPI_RECLAIMABLE</code> . . . . .	242
4.47.2.18 <code>LIMINE_MEMMAP_BAD_MEMORY</code> . . . . .	243
4.47.2.19 <code>LIMINE_MEMMAP_BOOTLOADER_RECLAIMABLE</code> . . . . .	243
4.47.2.20 <code>LIMINE_MEMMAP_FRAMEBUFFER</code> . . . . .	243
4.47.2.21 <code>LIMINE_MEMMAP_KERNEL_AND_MODULES</code> . . . . .	243

---

4.47.2.22 LIMINE_MEMMAP_REQUEST . . . . .	243
4.47.2.23 LIMINE_MEMMAP_RESERVED . . . . .	243
4.47.2.24 LIMINE_MEMMAP_USABLE . . . . .	244
4.47.2.25 LIMINE_MODULE_REQUEST . . . . .	244
4.47.2.26 LIMINE_PTR . . . . .	244
4.47.2.27 LIMINE_RSDP_REQUEST . . . . .	244
4.47.2.28 LIMINE_SMBIOS_REQUEST . . . . .	244
4.47.2.29 LIMINE_SMP_REQUEST . . . . .	244
4.47.2.30 LIMINE_STACK_SIZE_REQUEST . . . . .	245
4.47.2.31 LIMINE_TERMINAL_CB_BELL . . . . .	245
4.47.2.32 LIMINE_TERMINAL_CB_DEC . . . . .	245
4.47.2.33 LIMINE_TERMINAL_CB_KBD_LEDS . . . . .	245
4.47.2.34 LIMINE_TERMINAL_CB_LINUX . . . . .	245
4.47.2.35 LIMINE_TERMINAL_CB_MODE . . . . .	245
4.47.2.36 LIMINE_TERMINAL_CB_POS_REPORT . . . . .	246
4.47.2.37 LIMINE_TERMINAL_CB_PRIVATE_ID . . . . .	246
4.47.2.38 LIMINE_TERMINAL_CB_STATUS_REPORT . . . . .	246
4.47.2.39 LIMINE_TERMINAL_CTX_RESTORE . . . . .	246
4.47.2.40 LIMINE_TERMINAL_CTX_SAVE . . . . .	246
4.47.2.41 LIMINE_TERMINAL_CTX_SIZE . . . . .	246
4.47.2.42 LIMINE_TERMINAL_FULL_REFRESH . . . . .	247
4.47.2.43 LIMINE_TERMINAL_REQUEST . . . . .	247
4.47.3 Typedef Documentation . . . . .	247
4.47.3.1 limine_entry_point . . . . .	247
4.47.3.2 limine_goto_address . . . . .	247
4.47.3.3 limine_terminal_callback . . . . .	247
4.47.3.4 limine_terminal_write . . . . .	247
4.48 limine.h . . . . .	248
4.49 src/include/lock.h File Reference . . . . .	252
4.49.1 Data Structure Documentation . . . . .	254
4.49.1.1 struct smartlock . . . . .	254
4.49.2 Macro Definition Documentation . . . . .	254
4.49.2.1 _LOCK_C_H . . . . .	254
4.49.2.2 SMARTLOCK_INIT . . . . .	254
4.49.2.3 SPINLOCK_INIT . . . . .	255
4.49.3 Typedef Documentation . . . . .	255
4.49.3.1 spinlock_t . . . . .	255
4.49.4 Function Documentation . . . . .	255
4.49.4.1 atomic_lock() . . . . .	255
4.49.4.2 atomic_unlock() . . . . .	256
4.49.4.3 smartlock_acquire() . . . . .	256
4.49.4.4 smartlock_release() . . . . .	256

---

4.49.4.5 spinlock_acquire()	256
4.49.4.6 spinlock_release()	257
4.49.4.7 spinlock_test_and_acq()	257
4.50 lock.h	258
4.51 src/include/lock.hpp File Reference	258
4.52 lock.hpp	259
4.53 src/include/memUtils.h File Reference	260
4.53.1 Macro Definition Documentation	261
4.53.1.1 ALLOC	261
4.53.1.2 BSIZE	261
4.53.1.3 MEMUTILS_H	261
4.53.1.4 SIZE	261
4.53.2 Variable Documentation	261
4.53.2.1 kheap	262
4.54 memUtils.h	262
4.55 src/include/paging/frameallocator.h File Reference	262
4.55.1 Macro Definition Documentation	263
4.55.1.1 FRAME_EXPORT	263
4.55.1.2 PAGE_SIZE	263
4.55.2 Function Documentation	263
4.55.2.1 frame_free()	264
4.55.2.2 frame_free_multiple()	264
4.55.2.3 frame_lock()	265
4.55.2.4 frame_lock_multiple()	266
4.55.2.5 frame_request()	266
4.55.2.6 frame_request_multiple()	267
4.55.2.7 free_ram()	267
4.55.2.8 print_frame_bitmap()	268
4.55.2.9 read_memory_map()	268
4.55.2.10 reserved_ram()	269
4.55.2.11 used_ram()	269
4.56 frameallocator.h	270
4.57 src/include/paging/paging.h File Reference	270
4.57.1 Data Structure Documentation	271
4.57.1.1 struct PageTableOffset	271
4.57.2 Macro Definition Documentation	272
4.57.2.1 HIGHER_HALF_KERNEL_MEMORY_OFFSET	272
4.57.2.2 HIGHER_HALF_MEMORY_OFFSET	272
4.57.2.3 PAGE_ADDRESS_MASK	272
4.57.2.4 PAGE_FLAG_MASK	272
4.57.2.5 PAGING_EXPORT	272
4.57.3 Enumeration Type Documentation	272

---

4.57.3.1 PagingFlag . . . . .	272
4.57.4 Function Documentation . . . . .	273
4.57.4.1 __attribute__() . . . . .	273
4.57.4.2 IsHigherHalf() . . . . .	273
4.57.4.3 IsKernelHigherHalf() . . . . .	273
4.57.4.4 PagingDuplicate() . . . . .	274
4.57.4.5 PagingGetFreeFrame() . . . . .	274
4.57.4.6 PagingIdentityMap() . . . . .	275
4.57.4.7 PagingMapMemory() . . . . .	275
4.57.4.8 PagingPhysicalMemory() . . . . .	276
4.57.4.9 PagingSetActivePageTable() . . . . .	276
4.57.4.10 PagingUnmapMemory() . . . . .	277
4.57.4.11 TranslateToHighHalfMemoryAddress() . . . . .	277
4.57.4.12 TranslateToKernelMemoryAddress() . . . . .	278
4.57.4.13 TranslateToKernelPhysicalMemoryAddress() . . . . .	278
4.57.4.14 TranslateToPhysicalMemoryAddress() . . . . .	278
4.58 paging.h . . . . .	278
4.59 src/include/paging/vmm.h File Reference . . . . .	279
4.59.1 Data Structure Documentation . . . . .	280
4.59.1.1 struct dummy_proc . . . . .	280
4.59.2 Function Documentation . . . . .	281
4.59.2.1 VMM_table_clone() . . . . .	281
4.60 vmm.h . . . . .	281
4.61 src/include/pic.h File Reference . . . . .	282
4.61.1 Macro Definition Documentation . . . . .	283
4.61.1.1 PIC_EOI . . . . .	283
4.61.1.2 PIC_ICW1_ICW4 . . . . .	283
4.61.1.3 PIC_ICW1_INIT . . . . .	283
4.61.1.4 PIC_ICW1_INTERVAL4 . . . . .	283
4.61.1.5 PIC_ICW1_LEVEL . . . . .	284
4.61.1.6 PIC_ICW1_SINGLE . . . . .	284
4.61.1.7 PIC_ICW4_8086 . . . . .	284
4.61.1.8 PIC_ICW4_AUTO . . . . .	284
4.61.1.9 PIC_ICW4_BUF_MASTER . . . . .	284
4.61.1.10 PIC_ICW4_BUF_SLAVE . . . . .	284
4.61.1.11 PIC_MASTER . . . . .	285
4.61.1.12 PIC_MASTER_COMMAND . . . . .	285
4.61.1.13 PIC_MASTER_DATA . . . . .	285
4.61.1.14 PIC_SLAVE . . . . .	285
4.61.1.15 PIC_SLAVE_COMMAND . . . . .	285
4.61.1.16 PIC_SLAVE_DATA . . . . .	285
4.61.2 Function Documentation . . . . .	286

---

4.61.2.1 pic_enable()	286
4.61.2.2 pic_mask_irq()	286
4.61.2.3 pic_remap_offsets()	287
4.61.2.4 pic_send_eoi()	288
4.61.2.5 pic_unmask_irq()	289
4.62 pic.h	289
4.63 src/include/printf.h File Reference	290
4.63.1 Macro Definition Documentation	290
4.63.1.1 _PRINTF_H_	291
4.63.1.2 printf	291
4.63.1.3 snprintf	291
4.63.1.4 sprintf	292
4.63.1.5 vprintf	292
4.63.1.6 vsnprintf	292
4.63.2 Function Documentation	292
4.63.2.1 _putchar()	293
4.63.2.2 fctprintf()	294
4.63.2.3 printf_()	295
4.63.2.4 snprintf_()	297
4.63.2.5 sprintf_()	298
4.63.2.6 vprintf_()	299
4.63.2.7 vsnprintf_()	299
4.64 printf.h	300
4.65 src/include/proc.h File Reference	300
4.65.1 Data Structure Documentation	301
4.65.1.1 struct proc_vmm_map	301
4.65.1.2 struct process	302
4.65.2 Function Documentation	302
4.65.2.1 proc_yield()	302
4.66 proc.h	302
4.67 src/include/registers.h File Reference	303
4.67.1 Function Documentation	304
4.67.1.1 readCR2()	304
4.67.1.2 readCR3()	304
4.67.1.3 readCR4()	304
4.67.1.4 readCRO()	305
4.67.1.5 readRSP()	305
4.67.1.6 writeCR0()	305
4.67.1.7 writeCR3()	305
4.67.1.8 writeCR4()	306
4.67.1.9 writeMSR()	306
4.68 registers.h	306

---

4.69 src/include/sched.h File Reference . . . . .	307
4.70 sched.h . . . . .	307
4.71 src/include/serial.h File Reference . . . . .	307
4.71.1 Function Documentation . . . . .	307
4.71.1.1 serial_debug() . . . . .	307
4.71.1.2 serial_print() . . . . .	308
4.71.1.3 serial_print_line() . . . . .	308
4.72 serial.h . . . . .	308
4.73 src/include/shell.h File Reference . . . . .	309
4.73.1 Macro Definition Documentation . . . . .	309
4.73.1.1 _SHELL_H . . . . .	309
4.73.1.2 VSH_CMD_BUFFER_SIZE . . . . .	309
4.73.1.3 VSH_VERSION . . . . .	310
4.73.2 Function Documentation . . . . .	310
4.73.2.1 vsh_loop() . . . . .	310
4.73.2.2 vsh_readline() . . . . .	311
4.74 shell.h . . . . .	311
4.75 src/include/string.h File Reference . . . . .	312
4.75.1 Function Documentation . . . . .	312
4.75.1.1 memcmp() . . . . .	312
4.75.1.2 memcpy() . . . . .	313
4.75.1.3 memmove() . . . . .	313
4.75.1.4 memset() . . . . .	313
4.75.1.5 strlen() . . . . .	314
4.75.1.6 strtok() . . . . .	314
4.76 string.h . . . . .	314
4.77 src/include/terminal/framebuffer.h File Reference . . . . .	315
4.77.1 Data Structure Documentation . . . . .	316
4.77.1.1 struct fbterm_char . . . . .	316
4.77.1.2 struct fbterm_queue_item . . . . .	316
4.77.1.3 struct fbterm_context . . . . .	317
4.77.2 Macro Definition Documentation . . . . .	318
4.77.2.1 FBTERM_FONT_GLYPHS . . . . .	318
4.77.3 Function Documentation . . . . .	318
4.77.3.1 fbterm_init() . . . . .	319
4.78 framebuffer.h . . . . .	320
4.79 src/include/terminal/term.h File Reference . . . . .	322
4.79.1 Macro Definition Documentation . . . . .	323
4.79.1.1 TERM_CB_BELL . . . . .	323
4.79.1.2 TERM_CB_DEC . . . . .	323
4.79.1.3 TERM_CB_KBD_LEDS . . . . .	323
4.79.1.4 TERM_CB_LINUX . . . . .	323

---

4.79.1.5 TERM_CB_MODE . . . . .	323
4.79.1.6 TERM_CB_POS_REPORT . . . . .	324
4.79.1.7 TERM_CB_PRIVATE_ID . . . . .	324
4.79.1.8 TERM_CB_STATUS_REPORT . . . . .	324
4.79.1.9 TERM_MAX_ESC_VALUES . . . . .	324
4.79.2 Function Documentation . . . . .	324
4.79.2.1 term_context_reinit() . . . . .	325
4.79.2.2 term_write() . . . . .	325
4.80 term.h . . . . .	327
4.81 src/include/time.h File Reference . . . . .	329
4.81.1 Macro Definition Documentation . . . . .	329
4.81.1.1 _TIME_H . . . . .	329
4.81.2 Function Documentation . . . . .	330
4.81.2.1 init_PIT() . . . . .	330
4.81.2.2 print_date() . . . . .	330
4.81.2.3 print_sys_time() . . . . .	331
4.81.2.4 read_rtc() . . . . .	332
4.81.2.5 sleep() . . . . .	332
4.81.2.6 sys_clock_handler() . . . . .	333
4.81.3 Variable Documentation . . . . .	334
4.81.3.1 day . . . . .	334
4.81.3.2 hour . . . . .	334
4.81.3.3 minute . . . . .	334
4.81.3.4 month . . . . .	334
4.81.3.5 pit_armed . . . . .	335
4.81.3.6 second . . . . .	335
4.81.3.7 system_timer_fractions . . . . .	335
4.81.3.8 system_timer_ms . . . . .	335
4.81.3.9 year . . . . .	335
4.82 time.h . . . . .	335
4.83 src/include/tss.h File Reference . . . . .	336
4.83.1 Data Structure Documentation . . . . .	336
4.83.1.1 struct TSS . . . . .	336
4.83.2 Macro Definition Documentation . . . . .	337
4.83.2.1 _TSS_H . . . . .	337
4.84 tss.h . . . . .	337
4.85 src/include/vgafont.h File Reference . . . . .	338
4.85.1 Variable Documentation . . . . .	338
4.85.1.1 vgafont . . . . .	338
4.86 vgafont.h . . . . .	339
4.87 src/interrupts.c File Reference . . . . .	342
4.87.1 Function Documentation . . . . .	342

---

4.87.1.1 irq_handler() . . . . .	343
4.87.1.2 isr_exception_handler() . . . . .	343
4.87.2 Variable Documentation . . . . .	344
4.87.2.1 exception_messages . . . . .	344
4.87.2.2 irq_messages . . . . .	344
4.88 interrupts.c . . . . .	345
4.89 src/io.c File Reference . . . . .	346
4.89.1 Function Documentation . . . . .	347
4.89.1.1 inb() . . . . .	347
4.89.1.2 io_wait() . . . . .	347
4.89.1.3 outb() . . . . .	348
4.90 io.c . . . . .	348
4.91 src/kernel.c File Reference . . . . .	348
4.91.1 Macro Definition Documentation . . . . .	350
4.91.1.1 Black . . . . .	350
4.91.1.2 Blue . . . . .	350
4.91.1.3 Cyan . . . . .	350
4.91.1.4 Green . . . . .	350
4.91.1.5 Purple . . . . .	351
4.91.1.6 Red . . . . .	351
4.91.1.7 White . . . . .	351
4.91.1.8 Yellow . . . . .	351
4.91.2 Function Documentation . . . . .	351
4.91.2.1 _start() . . . . .	352
4.91.2.2 breakpoint() . . . . .	353
4.91.2.3 clear_screen() . . . . .	353
4.91.2.4 halt() . . . . .	354
4.91.2.5 print_prompt() . . . . .	354
4.91.2.6 start_interrupts() . . . . .	355
4.91.2.7 stop_interrupts() . . . . .	355
4.91.2.8 task_switch_int() . . . . .	355
4.91.3 Variable Documentation . . . . .	355
4.91.3.1 bootspace . . . . .	356
4.91.3.2 early_term . . . . .	356
4.91.3.3 Kaddress_req . . . . .	356
4.91.3.4 kerror_mode . . . . .	356
4.91.3.5 kheap . . . . .	356
4.91.3.6 term_bg . . . . .	357
4.91.3.7 term_context . . . . .	357
4.91.3.8 term_fg . . . . .	357
4.91.3.9 test_table . . . . .	357
4.92 kernel.c . . . . .	357

---

4.93 src/KernelUtils.c File Reference . . . . .	360
4.93.1 Macro Definition Documentation . . . . .	361
4.93.1.1 ALIGN_DOWN . . . . .	361
4.93.1.2 ALIGN_UP . . . . .	361
4.93.2 Typedef Documentation . . . . .	361
4.93.2.1 symbol . . . . .	362
4.93.3 Function Documentation . . . . .	362
4.93.3.1 breakpoint() . . . . .	362
4.93.3.2 get_memory_size() . . . . .	362
4.93.3.3 halt() . . . . .	363
4.93.3.4 init_memory() . . . . .	363
4.93.3.5 print_memmap() . . . . .	364
4.93.3.6 print_memory() . . . . .	364
4.93.3.7 system_readout() . . . . .	365
4.93.4 Variable Documentation . . . . .	366
4.93.4.1 data_end_addr . . . . .	366
4.93.4.2 data_start_addr . . . . .	366
4.93.4.3 fbr_req . . . . .	366
4.93.4.4 frameBitmap . . . . .	366
4.93.4.5 k_mode . . . . .	367
4.93.4.6 Kaddress_req . . . . .	367
4.93.4.7 memmap_req . . . . .	367
4.93.4.8 page_table . . . . .	367
4.93.4.9 rodata_end_addr . . . . .	367
4.93.4.10 rodata_start_addr . . . . .	368
4.93.4.11 smp_req . . . . .	368
4.93.4.12 text_end_addr . . . . .	368
4.93.4.13 text_start_addr . . . . .	368
4.94 KernelUtils.c . . . . .	368
4.95 src/liballoc.c File Reference . . . . .	372
4.95.1 Macro Definition Documentation . . . . .	374
4.95.1.1 LIBALLOC_MAGIC . . . . .	374
4.95.1.2 MAXCOMPLETE . . . . .	374
4.95.1.3 MAXEXP . . . . .	374
4.95.1.4 MINEXP . . . . .	374
4.95.1.5 MODE . . . . .	374
4.95.1.6 MODE_BEST . . . . .	375
4.95.1.7 MODE_INSTANT . . . . .	375
4.95.2 Function Documentation . . . . .	375
4.95.2.1 absorb_right() . . . . .	375
4.95.2.2 allocate_new_tag() . . . . .	376
4.95.2.3 calloc() . . . . .	376

---

4.95.2.4 free()	377
4.95.2.5 getexp()	377
4.95.2.6 insert_tag()	378
4.95.2.7 liballoc_memcpy()	379
4.95.2.8 liballoc_memset()	379
4.95.2.9 malloc()	380
4.95.2.10 melt_left()	380
4.95.2.11 realloc()	381
4.95.2.12 remove_tag()	381
4.95.2.13 split_tag()	382
4.95.3 Variable Documentation	382
4.95.3.1 l_completePages	382
4.95.3.2 l_freePages	383
4.95.3.3 l_initialized	383
4.95.3.4 l_pageCount	383
4.95.3.5 lPageSize	383
4.96 liballoc.c	383
4.97 src/lock.cpp File Reference	389
4.98 lock.cpp	390
4.99 src/lock_atm_c.cpp File Reference	390
4.99.1 Function Documentation	391
4.99.1.1 atomic_lock()	391
4.99.1.2 atomic_unlock()	392
4.99.2 Variable Documentation	392
4.99.2.1 c_lock	392
4.100 lock_atm_c.cpp	392
4.101 src/lock_c.c File Reference	393
4.102 lock_c.c	393
4.103 src/memcmp.c File Reference	393
4.103.1 Function Documentation	394
4.103.1.1 memcmp()	394
4.104 memcmp.c	394
4.105 src/memcpy.c File Reference	394
4.105.1 Function Documentation	395
4.105.1.1 memcpy()	395
4.106 memcpy.c	395
4.107 src/memmove.c File Reference	395
4.107.1 Function Documentation	396
4.107.1.1 memmove()	396
4.108 memmove.c	396
4.109 src/memset.c File Reference	396
4.109.1 Function Documentation	397

---

4.109.1.1 <code>memset()</code>	397
4.110 <code>memset.c</code>	397
4.111 <code>src/memUtils.c</code> File Reference	398
4.111.1 Function Documentation	398
4.111.1.1 <code>PagingGetFreeFrame()</code>	398
4.112 <code>memUtils.c</code>	399
4.113 <code>src/paging.c</code> File Reference	399
4.113.1 Function Documentation	400
4.113.1.1 <code>DuplicateRecursive()</code>	400
4.113.1.2 <code>GetOrAllocEntry()</code>	401
4.113.1.3 <code>GetOrNullifyEntry()</code>	401
4.113.1.4 <code>OffsetToVirtualAddress()</code>	402
4.113.1.5 <code>PagingDuplicate()</code>	402
4.113.1.6 <code>PagingIdentityMap()</code>	403
4.113.1.7 <code>PagingMapMemory()</code>	403
4.113.1.8 <code>PagingPhysicalMemory()</code>	404
4.113.1.9 <code>PagingUnmapMemory()</code>	404
4.113.1.10 <code>ReadCR3()</code>	405
4.113.1.11 <code>VirtualAddressToOffsets()</code>	405
4.113.1.12 <code>WriteCR3()</code>	405
4.114 <code>paging.c</code>	406
4.115 <code>src/pic.c</code> File Reference	408
4.115.1 Function Documentation	409
4.115.1.1 <code>pic_enable()</code>	409
4.115.1.2 <code>pic_mask_irq()</code>	410
4.115.1.3 <code>pic_remap_offsets()</code>	410
4.115.1.4 <code>pic_send_eoi()</code>	411
4.115.1.5 <code>pic_unmask_irq()</code>	412
4.116 <code>pic.c</code>	412
4.117 <code>src/printf.c</code> File Reference	413
4.117.1 Macro Definition Documentation	415
4.117.1.1 <code>FLAGS_ADAPT_EXP</code>	415
4.117.1.2 <code>FLAGS_CHAR</code>	415
4.117.1.3 <code>FLAGS_HASH</code>	416
4.117.1.4 <code>FLAGS_LEFT</code>	416
4.117.1.5 <code>FLAGS_LONG</code>	416
4.117.1.6 <code>FLAGS_LONG_LONG</code>	416
4.117.1.7 <code>FLAGS_PLUS</code>	416
4.117.1.8 <code>FLAGS_PRECISION</code>	416
4.117.1.9 <code>FLAGS_SHORT</code>	417
4.117.1.10 <code>FLAGS_SPACE</code>	417
4.117.1.11 <code>FLAGS_UPPERCASE</code>	417

---

4.117.1.12 FLAGS_ZEROPAD . . . . .	417
4.117.1.13 PRNTF_DEFAULT_FLOAT_PRECISION . . . . .	417
4.117.1.14 PRNTF_FTOA_BUFFER_SIZE . . . . .	417
4.117.1.15 PRNTF_MAX_FLOAT . . . . .	418
4.117.1.16 PRNTF_NTOA_BUFFER_SIZE . . . . .	418
4.117.1.17 PRNTF_SUPPORT_EXPONENTIAL . . . . .	418
4.117.1.18 PRNTF_SUPPORT_FLOAT . . . . .	418
4.117.1.19 PRNTF_SUPPORT_LONG_LONG . . . . .	418
4.117.1.20 PRNTF_SUPPORT_PTRDIFF_T . . . . .	418
4.117.2 Typedef Documentation . . . . .	419
4.117.2.1 out_fct_type . . . . .	419
4.117.3 Function Documentation . . . . .	419
4.117.3.1 atoi() . . . . .	419
4.117.3.2 etoa() . . . . .	420
4.117.3.3 ftoa() . . . . .	422
4.117.3.4 is_digit() . . . . .	424
4.117.3.5 ntoa_format() . . . . .	425
4.117.3.6 ntoa_long() . . . . .	427
4.117.3.7 ntoa_long_long() . . . . .	429
4.117.3.8 _out_buffer() . . . . .	431
4.117.3.9 _out_char() . . . . .	432
4.117.3.10 _out_fct() . . . . .	433
4.117.3.11 _out_null() . . . . .	434
4.117.3.12 _out_rev() . . . . .	435
4.117.3.13 strnlen_s() . . . . .	437
4.117.3.14 vsnprintf() . . . . .	438
4.117.3.15 fctprintf() . . . . .	439
4.117.3.16 printf_() . . . . .	440
4.117.3.17 snprintf_() . . . . .	442
4.117.3.18 sprintf_() . . . . .	443
4.117.3.19 vprintf_() . . . . .	444
4.117.3.20 vsnprintf_() . . . . .	444
4.118 printf.c . . . . .	445
4.119 src/proc.c File Reference . . . . .	457
4.120 proc.c . . . . .	457
4.121 src/putchar.c File Reference . . . . .	457
4.121.1 Function Documentation . . . . .	458
4.121.1.1 putchar() . . . . .	458
4.122 putchar.c . . . . .	459
4.123 src/registers.c File Reference . . . . .	460
4.123.1 Function Documentation . . . . .	461
4.123.1.1 writeMSR() . . . . .	461

4.124 registers.c . . . . .	461
4.125 src/sched.c File Reference . . . . .	461
4.126 sched.c . . . . .	461
4.127 src/serial.c File Reference . . . . .	461
4.127.1 Function Documentation . . . . .	462
4.127.1.1 serial_print() . . . . .	462
4.127.1.2 serial_print_line() . . . . .	463
4.128 serial.c . . . . .	463
4.129 src/shell.c File Reference . . . . .	463
4.129.1 Function Documentation . . . . .	464
4.129.1.1 cmd_parser() . . . . .	464
4.129.1.2 vsh_loop() . . . . .	465
4.129.1.3 vsh_readline() . . . . .	465
4.130 shell.c . . . . .	466
4.131 src/strlen.c File Reference . . . . .	467
4.131.1 Function Documentation . . . . .	468
4.131.1.1 strlen() . . . . .	468
4.132 strlen.c . . . . .	469
4.133 src/strtok.c File Reference . . . . .	469
4.133.1 Function Documentation . . . . .	469
4.133.1.1 strtok() . . . . .	469
4.134 strtok.c . . . . .	470
4.135 src/terminal/framebuffer.c File Reference . . . . .	471
4.135.1 Macro Definition Documentation . . . . .	473
4.135.1.1 FONT_BYTES . . . . .	473
4.135.2 Function Documentation . . . . .	473
4.135.2.1 compare_char() . . . . .	473
4.135.2.2 draw_cursor() . . . . .	474
4.135.2.3 fbterm_clear() . . . . .	474
4.135.2.4 fbterm_deinit() . . . . .	475
4.135.2.5 fbterm_disable_cursor() . . . . .	475
4.135.2.6 fbterm_double_buffer_flush() . . . . .	476
4.135.2.7 fbterm_enable_cursor() . . . . .	476
4.135.2.8 fbterm_full_refresh() . . . . .	477
4.135.2.9 fbterm_get_cursor_pos() . . . . .	477
4.135.2.10 fbterm_init() . . . . .	478
4.135.2.11 fbterm_move_character() . . . . .	479
4.135.2.12 fbterm_raw_putchar() . . . . .	480
4.135.2.13 fbterm_restore_state() . . . . .	481
4.135.2.14 fbterm_revscroll() . . . . .	481
4.135.2.15 fbterm_save_state() . . . . .	482
4.135.2.16 fbterm_scroll() . . . . .	482

4.135.2.17 fbterm_set_cursor_pos()	483
4.135.2.18 fbterm_set_text_bg()	483
4.135.2.19 fbterm_set_text_bg_bright()	484
4.135.2.20 fbterm_set_text_bg_default()	484
4.135.2.21 fbterm_set_text_bg_rgb()	485
4.135.2.22 fbterm_set_text_fg()	485
4.135.2.23 fbterm_set_text_fg_bright()	486
4.135.2.24 fbterm_set_text_fg_default()	486
4.135.2.25 fbterm_set_text_fg_rgb()	487
4.135.2.26 fbterm_swap_palette()	487
4.135.2.27 memcpy()	488
4.135.2.28 memset()	488
4.135.2.29 plot_char()	489
4.135.2.30 plot_char_fast()	489
4.135.2.31 push_to_queue()	490
4.135.3 Variable Documentation	490
4.135.3.1 builtin_font	490
4.136 framebuffer.c	491
4.137 src/terminal/term.c File Reference	500
4.137.1 Macro Definition Documentation	501
4.137.1.1 CHARSET_DEC_SPECIAL	502
4.137.1.2 CHARSET_DEFAULT	502
4.137.2 Function Documentation	502
4.137.2.1 control_sequence_parse()	502
4.137.2.2 dec_private_parse()	503
4.137.2.3 dec_special_to_cp437()	504
4.137.2.4 escape_parse()	505
4.137.2.5 linux_private_parse()	507
4.137.2.6 mode_toggle()	508
4.137.2.7 restore_state()	509
4.137.2.8 save_state()	510
4.137.2.9 sgr()	511
4.137.2.10 term_context_reinit()	512
4.137.2.11 term_putchar()	513
4.137.2.12 term_write()	515
4.137.3 Variable Documentation	516
4.137.3.1 col256	516
4.138 term.c	517
4.139 src/time.c File Reference	529
4.139.1 Macro Definition Documentation	530
4.139.1.1 CHANNEL_ZERO	530
4.139.1.2 CMOS_ADDR	530

---

4.139.1.3 CMOS_DATA . . . . .	530
4.139.1.4 CURRENT_YEAR . . . . .	530
4.139.1.5 MODE_CMD . . . . .	531
4.139.2 Function Documentation . . . . .	531
4.139.2.1 config_PIT() . . . . .	531
4.139.2.2 dayofweek() . . . . .	531
4.139.2.3 delta_int() . . . . .	532
4.139.2.4 get_RTC_register() . . . . .	532
4.139.2.5 get_update_flag() . . . . .	532
4.139.2.6 init_PIT() . . . . .	533
4.139.2.7 print_date() . . . . .	534
4.139.2.8 print_sys_time() . . . . .	535
4.139.2.9 read_rtc() . . . . .	535
4.139.2.10 sleep() . . . . .	536
4.139.2.11 sys_clock_handler() . . . . .	537
4.139.2.12 task_switch_handler() . . . . .	537
4.139.2.13 timer_irq() . . . . .	538
4.139.3 Variable Documentation . . . . .	538
4.139.3.1 century_register . . . . .	538
4.139.3.2 count_down . . . . .	538
4.139.3.3 day . . . . .	539
4.139.3.4 hour . . . . .	539
4.139.3.5 minute . . . . .	539
4.139.3.6 month . . . . .	539
4.139.3.7 month_str . . . . .	539
4.139.3.8 pit_armed . . . . .	540
4.139.3.9 second . . . . .	540
4.139.3.10 system_timer_fractions . . . . .	540
4.139.3.11 system_timer_ms . . . . .	540
4.139.3.12 task_timer_count . . . . .	540
4.139.3.13 weekday . . . . .	540
4.139.3.14 year . . . . .	541
4.140 time.c . . . . .	541
4.141 src/vmm.c File Reference . . . . .	544
4.141.1 Function Documentation . . . . .	544
4.141.1.1 VMM_table_clone() . . . . .	545
4.141.2 Variable Documentation . . . . .	545
4.141.2.1 proc_page_size . . . . .	545
4.141.2.2 test_proc . . . . .	545
4.141.2.3 vmm_page_table . . . . .	545
4.142 vmm.c . . . . .	546





# Chapter 1

## Data Structure Index

### 1.1 Data Structures

Here are the data structures with brief descriptions:

AtomicLock . . . . .	5
cpu_local . . . . .	7
limine_5_level_paging_request . . . . .	9
limine_boot_time_request . . . . .	10
limine_bootloader_info_request . . . . .	11
limine_bootloader_info_response . . . . .	12
limine_dtb_request . . . . .	13
limine_dtb_response . . . . .	14
limine_efi_system_table_request . . . . .	14
limine_efi_system_table_response . . . . .	16
limine_entry_point_request . . . . .	16
limine_file . . . . .	18
limine_framebuffer . . . . .	21
limine_framebuffer_request . . . . .	24
limine_framebuffer_response . . . . .	25
limine_hhdm_request . . . . .	26
limine_kernel_address_request . . . . .	28
limine_kernel_file_request . . . . .	29
limine_kernel_file_response . . . . .	30
limine_memmap_request . . . . .	30
limine_memmap_response . . . . .	32
limine_module_request . . . . .	33
limine_module_response . . . . .	34
limine_rsdp_request . . . . .	35
limine_rsdp_response . . . . .	36
limine_smbios_request . . . . .	36
limine_smbios_response . . . . .	38
limine_smp_request . . . . .	39
limine_stack_size_request . . . . .	40
limine_terminal . . . . .	41
limine_terminal_request . . . . .	42
limine_terminal_response . . . . .	43
out_fct_wrap_type . . . . .	45
ScopedLock . . . . .	46
term_context . . . . .	47



# Chapter 2

## File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

src/alloc.cpp . . . . .	59
src/cpudet-clean.c . . . . .	63
src/cpuUtils.c . . . . .	72
src/frameallocator.c . . . . .	109
src/gdt.c . . . . .	122
src/heap.c . . . . .	128
src/idt.c . . . . .	135
src/interrupts.c . . . . .	342
src/io.c . . . . .	346
src/kernel.c . . . . .	348
src/KernelUtils.c . . . . .	360
src/liballoc.c . . . . .	372
src/lock.cpp . . . . .	389
src/lock_atm_c.cpp . . . . .	390
src/lock_c.c . . . . .	393
src/memcmp.c . . . . .	393
src/memcpy.c . . . . .	394
src/memmove.c . . . . .	395
src/memset.c . . . . .	396
src/memUtils.c . . . . .	398
src/paging.c . . . . .	399
src/pic.c . . . . .	408
src/printf.c . . . . .	413
src/proc.c . . . . .	457
src/putchar.c . . . . .	457
src/registers.c . . . . .	460
src/sched.c . . . . .	461
src/serial.c . . . . .	461
src/shell.c . . . . .	463
src/strlen.c . . . . .	467
src/strtok.c . . . . .	469
src/time.c . . . . .	529
src/vmm.c . . . . .	544
src/drivers/keyboard/keyboard.c . . . . .	99
src/include/bitmap.h . . . . .	140

src/include/cpuUtils.h . . . . .	142
src/include/gdt.h . . . . .	172
src/include/global_defs.h . . . . .	180
src/include/heap.h . . . . .	183
src/include/idt.h . . . . .	189
src/include/interrupts.h . . . . .	196
src/include/io.h . . . . .	199
src/include/kernel.h . . . . .	202
src/include/KernelUtils.h . . . . .	205
src/include/liballoc.h . . . . .	228
src/include/limine.h . . . . .	235
src/include/lock.h . . . . .	252
src/include/lock.hpp . . . . .	258
src/include/memUtils.h . . . . .	260
src/include/pic.h . . . . .	282
src/include/printf.h . . . . .	290
src/include/proc.h . . . . .	300
src/include/registers.h . . . . .	303
src/include/sched.h . . . . .	307
src/include/serial.h . . . . .	307
src/include/shell.h . . . . .	309
src/include/string.h . . . . .	312
src/include/time.h . . . . .	329
src/include/tss.h . . . . .	336
src/include/vgafont.h . . . . .	338
src/include/drivers/keyboard/keyboard.h . . . . .	161
src/include/drivers/keyboard/keyboard_map.h . . . . .	167
src/include/lib/hashmap.h . . . . .	212
src/include/lib/vector.h . . . . .	220
src/include/paging/frameallocator.h . . . . .	262
src/include/paging/paging.h . . . . .	270
src/include/paging/vmm.h . . . . .	279
src/include/terminal/framebuffer.h . . . . .	315
src/include/terminal/term.h . . . . .	322
src/terminal/framebuffer.c . . . . .	471
src/terminal/term.c . . . . .	500

## Chapter 3

# Data Structure Documentation

### 3.1 AtomicLock Class Reference

```
#include <lock.hpp>
```

#### Public Member Functions

- `AtomicLock ()`
- `bool IsLocked () const`
- `void Lock ()`
- `void Unlock ()`
- `void ForceLock ()`

#### 3.1.1 Detailed Description

Definition at line [6](#) of file `lock.hpp`.

#### 3.1.2 Constructor & Destructor Documentation

##### 3.1.2.1 AtomicLock()

```
AtomicLock::AtomicLock ( )
```

Definition at line [3](#) of file `lock.cpp`.

#### 3.1.3 Member Function Documentation

### 3.1.3.1 ForceLock()

```
void AtomicLock::ForceLock ( )
```

Definition at line 27 of file [lock.cpp](#).

### 3.1.3.2 IsLocked()

```
bool AtomicLock::IsLocked ( ) const
```

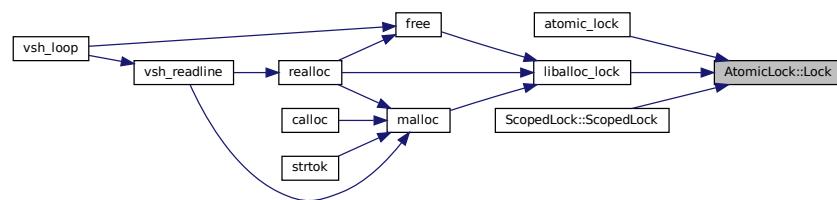
Definition at line 5 of file [lock.cpp](#).

### 3.1.3.3 Lock()

```
void AtomicLock::Lock ( )
```

Definition at line 14 of file [lock.cpp](#).

Here is the caller graph for this function:

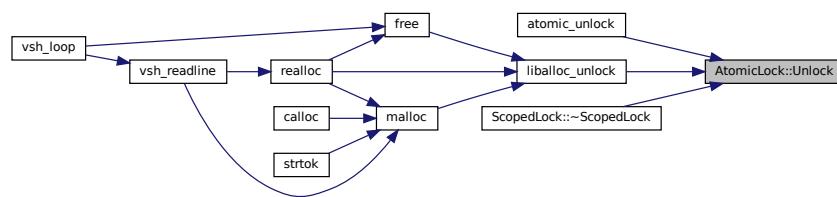


### 3.1.3.4 Unlock()

```
void AtomicLock::Unlock ( )
```

Definition at line 32 of file [lock.cpp](#).

Here is the caller graph for this function:



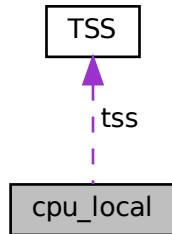
The documentation for this class was generated from the following files:

- [src/include/lock.hpp](#)
- [src/lock.cpp](#)

## 3.2 cpu\_local Struct Reference

```
#include <cpuUtils.h>
```

Collaboration diagram for cpu\_local:



### Data Fields

- int `cpu_number`
- bool `bsp`
- bool `active`
- int `last_run_queue_index`
- struct `TSS` `tss`
- struct `thread` \* `idle_thread`
- void(\* `timer_function` )(int, struct `cpu_ctx` \*)

### 3.2.1 Detailed Description

Definition at line 76 of file [cpuUtils.h](#).

### 3.2.2 Field Documentation

#### 3.2.2.1 active

```
bool cpu_local::active
```

Definition at line 81 of file [cpuUtils.h](#).

### 3.2.2.2 bsp

```
bool cpu_local::bsp
```

Definition at line 80 of file [cpuUtils.h](#).

### 3.2.2.3 cpu\_number

```
int cpu_local::cpu_number
```

Definition at line 79 of file [cpuUtils.h](#).

### 3.2.2.4 idle\_thread

```
struct thread* cpu_local::idle_thread
```

Definition at line 84 of file [cpuUtils.h](#).

### 3.2.2.5 last\_run\_queue\_index

```
int cpu_local::last_run_queue_index
```

Definition at line 82 of file [cpuUtils.h](#).

### 3.2.2.6 timer\_function

```
void(* cpu_local::timer_function) (int, struct cpu_ctx *)
```

Definition at line 85 of file [cpuUtils.h](#).

### 3.2.2.7 tss

```
struct TSS cpu_local::tss
```

Definition at line 82 of file [cpuUtils.h](#).

The documentation for this struct was generated from the following file:

- [src/include/cpuUtils.h](#)

### 3.3 limine\_5\_level\_paging\_request Struct Reference

```
#include <limine.h>
```

#### Public Member Functions

- `LIMINE_PTR (struct limine_5_level_paging_response *) response`

#### Data Fields

- `uint64_t id [4]`
- `uint64_t revision`

#### 3.3.1 Detailed Description

Definition at line 181 of file [limine.h](#).

#### 3.3.2 Member Function Documentation

##### 3.3.2.1 LIMINE\_PTR()

```
limine_5_level_paging_request::LIMINE_PTR (
    struct limine_5_level_paging_response * )
```

#### 3.3.3 Field Documentation

##### 3.3.3.1 id

```
uint64_t limine_5_level_paging_request::id[4]
```

Definition at line 182 of file [limine.h](#).

##### 3.3.3.2 revision

```
uint64_t limine_5_level_paging_request::revision
```

Definition at line 183 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- `src/include/limine.h`

## 3.4 limine\_boot\_time\_request Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- `LIMINE_PTR (struct limine_boot_time_response *) response`

### Data Fields

- `uint64_t id [4]`
- `uint64_t revision`

#### 3.4.1 Detailed Description

Definition at line 379 of file [limine.h](#).

#### 3.4.2 Member Function Documentation

##### 3.4.2.1 LIMINE\_PTR()

```
limine_boot_time_request::LIMINE_PTR (
    struct limine_boot_time_response * )
```

#### 3.4.3 Field Documentation

##### 3.4.3.1 id

```
uint64_t limine_boot_time_request::id[4]
```

Definition at line 380 of file [limine.h](#).

##### 3.4.3.2 revision

```
uint64_t limine_boot_time_request::revision
```

Definition at line 381 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- `src/include/limine.h`

## 3.5 limine\_bootloader\_info\_request Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- `LIMINE_PTR (struct limine_bootloader_info_response *) response`

### Data Fields

- `uint64_t id [4]`
- `uint64_t revision`

#### 3.5.1 Detailed Description

Definition at line 58 of file [limine.h](#).

#### 3.5.2 Member Function Documentation

##### 3.5.2.1 LIMINE\_PTR()

```
limine_bootloader_info_request::LIMINE_PTR (
    struct limine_bootloader_info_response * )
```

#### 3.5.3 Field Documentation

##### 3.5.3.1 id

```
uint64_t limine_bootloader_info_request::id[4]
```

Definition at line 59 of file [limine.h](#).

##### 3.5.3.2 revision

```
uint64_t limine_bootloader_info_request::revision
```

Definition at line 60 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- `src/include/limine.h`

## 3.6 limine\_bootloader\_info\_response Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- [LIMINE\\_PTR \(char \\*\) name](#)
- [LIMINE\\_PTR \(char \\*\) version](#)

### Data Fields

- `uint64_t revision`

#### 3.6.1 Detailed Description

Definition at line 52 of file [limine.h](#).

#### 3.6.2 Member Function Documentation

##### 3.6.2.1 LIMINE\_PTR() [1/2]

```
limine_bootloader_info_response::LIMINE_PTR (
    char * )
```

##### 3.6.2.2 LIMINE\_PTR() [2/2]

```
limine_bootloader_info_response::LIMINE_PTR (
    char * )
```

#### 3.6.3 Field Documentation

##### 3.6.3.1 revision

```
uint64_t limine_bootloader_info_response::revision
```

Definition at line 53 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- `src/include/limine.h`

## 3.7 limine\_dtb\_request Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- `LIMINE_PTR (struct limine_dtb_response *) response`

### Data Fields

- `uint64_t id [4]`
- `uint64_t revision`

#### 3.7.1 Detailed Description

Definition at line 410 of file [limine.h](#).

#### 3.7.2 Member Function Documentation

##### 3.7.2.1 LIMINE\_PTR()

```
limine_dtb_request::LIMINE_PTR (
    struct limine_dtb_response * )
```

#### 3.7.3 Field Documentation

##### 3.7.3.1 id

```
uint64_t limine_dtb_request::id[4]
```

Definition at line 411 of file [limine.h](#).

##### 3.7.3.2 revision

```
uint64_t limine_dtb_request::revision
```

Definition at line 412 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- `src/include/limine.h`

## 3.8 limine\_dtb\_response Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- [LIMINE\\_PTR](#) (void \*) dtb\_ptr

### Data Fields

- uint64\_t revision

#### 3.8.1 Detailed Description

Definition at line [405](#) of file [limine.h](#).

#### 3.8.2 Member Function Documentation

##### 3.8.2.1 LIMINE\_PTR()

```
limine_dtb_response::LIMINE_PTR (
    void * )
```

#### 3.8.3 Field Documentation

##### 3.8.3.1 revision

```
uint64_t limine_dtb_response::revision
```

Definition at line [406](#) of file [limine.h](#).

The documentation for this struct was generated from the following file:

- src/include/[limine.h](#)

## 3.9 limine\_esi\_system\_table\_request Struct Reference

```
#include <limine.h>
```

## Public Member Functions

- LIMINE\_PTR (struct [limine\\_efi\\_system\\_table\\_response](#) \*) response

## Data Fields

- uint64\_t **id** [4]
- uint64\_t **revision**

### 3.9.1 Detailed Description

Definition at line 364 of file [limine.h](#).

### 3.9.2 Member Function Documentation

#### 3.9.2.1 LIMINE\_PTR()

```
limine_efi_system_table_request::LIMINE_PTR (
    struct limine\_efi\_system\_table\_response * )
```

### 3.9.3 Field Documentation

#### 3.9.3.1 id

```
uint64_t limine_efi_system_table_request::id[4]
```

Definition at line 365 of file [limine.h](#).

#### 3.9.3.2 revision

```
uint64_t limine_efi_system_table_request::revision
```

Definition at line 366 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- src/include/[limine.h](#)

## 3.10 limine\_efi\_system\_table\_response Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- [LIMINE\\_PTR](#) (void \*) address

### Data Fields

- uint64\_t [revision](#)

#### 3.10.1 Detailed Description

Definition at line [359](#) of file [limine.h](#).

#### 3.10.2 Member Function Documentation

##### 3.10.2.1 LIMINE\_PTR()

```
limine_efi_system_table_response::LIMINE_PTR (
    void * )
```

#### 3.10.3 Field Documentation

##### 3.10.3.1 revision

```
uint64_t limine_efi_system_table_response::revision
```

Definition at line [360](#) of file [limine.h](#).

The documentation for this struct was generated from the following file:

- src/include/[limine.h](#)

## 3.11 limine\_entry\_point\_request Struct Reference

```
#include <limine.h>
```

## Public Member Functions

- LIMINE\_PTR (struct limine\_entry\_point\_response \*) response
- LIMINE\_PTR (limine\_entry\_point) entry

## Data Fields

- uint64\_t id [4]
- uint64\_t revision

### 3.11.1 Detailed Description

Definition at line 286 of file [limine.h](#).

### 3.11.2 Member Function Documentation

#### 3.11.2.1 LIMINE\_PTR() [1/2]

```
limine_entry_point_request::LIMINE_PTR (
    limine_entry_point )
```

#### 3.11.2.2 LIMINE\_PTR() [2/2]

```
limine_entry_point_request::LIMINE_PTR (
    struct limine_entry_point_response * )
```

### 3.11.3 Field Documentation

#### 3.11.3.1 id

```
uint64_t limine_entry_point_request::id[4]
```

Definition at line 287 of file [limine.h](#).

### 3.11.3.2 revision

```
uint64_t limine_entry_point_request::revision
```

Definition at line 288 of file [limine.h](#).

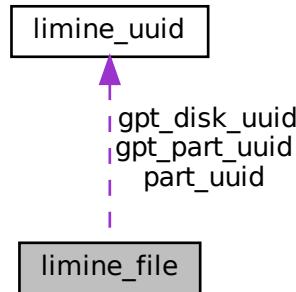
The documentation for this struct was generated from the following file:

- [src/include/limine.h](#)

## 3.12 limine\_file Struct Reference

```
#include <limine.h>
```

Collaboration diagram for limine\_file:



### Public Member Functions

- [LIMINE\\_PTR](#) (void \*) address
- [LIMINE\\_PTR](#) (char \*) path
- [LIMINE\\_PTR](#) (char \*) cmdline

### Data Fields

- uint64\_t [revision](#)
- uint64\_t [size](#)
- uint32\_t [media\\_type](#)
- uint32\_t [unused](#)
- uint32\_t [tftp\\_ip](#)
- uint32\_t [tftp\\_port](#)
- uint32\_t [partition\\_index](#)
- uint32\_t [mbr\\_disk\\_id](#)
- struct [limine\\_uuid gpt\\_disk\\_uuid](#)
- struct [limine\\_uuid gpt\\_part\\_uuid](#)
- struct [limine\\_uuid part\\_uuid](#)

### 3.12.1 Detailed Description

Definition at line 31 of file [limine.h](#).

### 3.12.2 Member Function Documentation

#### 3.12.2.1 LIMINE\_PTR() [1/3]

```
limine_file::LIMINE_PTR (
    char * )
```

#### 3.12.2.2 LIMINE\_PTR() [2/3]

```
limine_file::LIMINE_PTR (
    char * )
```

#### 3.12.2.3 LIMINE\_PTR() [3/3]

```
limine_file::LIMINE_PTR (
    void * )
```

### 3.12.3 Field Documentation

#### 3.12.3.1 gpt\_disk\_uuid

```
struct limine_uuid limine_file::gpt_disk_uuid
```

Definition at line 42 of file [limine.h](#).

#### 3.12.3.2 gpt\_part\_uuid

```
struct limine_uuid limine_file::gpt_part_uuid
```

Definition at line 42 of file [limine.h](#).

### 3.12.3.3 mbr\_disk\_id

```
uint32_t limine_file::mbr_disk_id
```

Definition at line [42](#) of file [limine.h](#).

### 3.12.3.4 media\_type

```
uint32_t limine_file::media_type
```

Definition at line [37](#) of file [limine.h](#).

### 3.12.3.5 part\_uuid

```
struct limine_uuid limine_file::part_uuid
```

Definition at line [42](#) of file [limine.h](#).

### 3.12.3.6 partition\_index

```
uint32_t limine_file::partition_index
```

Definition at line [41](#) of file [limine.h](#).

### 3.12.3.7 revision

```
uint64_t limine_file::revision
```

Definition at line [32](#) of file [limine.h](#).

### 3.12.3.8 size

```
uint64_t limine_file::size
```

Definition at line [34](#) of file [limine.h](#).

### 3.12.3.9 tftp\_ip

```
uint32_t limine_file::tftp_ip
```

Definition at line 39 of file [limine.h](#).

### 3.12.3.10 tftp\_port

```
uint32_t limine_file::tftp_port
```

Definition at line 40 of file [limine.h](#).

### 3.12.3.11 unused

```
uint32_t limine_file::unused
```

Definition at line 38 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- [src/include/limine.h](#)

## 3.13 limine\_framebuffer Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- [LIMINE\\_PTR \(void \\*\) address](#)
- [LIMINE\\_PTR \(void \\*\) edid](#)

### Data Fields

- [uint64\\_t width](#)
- [uint64\\_t height](#)
- [uint64\\_t pitch](#)
- [uint16\\_t bpp](#)
- [uint8\\_t memory\\_model](#)
- [uint8\\_t red\\_mask\\_size](#)
- [uint8\\_t red\\_mask\\_shift](#)
- [uint8\\_t green\\_mask\\_size](#)
- [uint8\\_t green\\_mask\\_shift](#)
- [uint8\\_t blue\\_mask\\_size](#)
- [uint8\\_t blue\\_mask\\_shift](#)
- [uint8\\_t unused \[7\]](#)
- [uint64\\_t edid\\_size](#)

### 3.13.1 Detailed Description

Definition at line 100 of file [limine.h](#).

### 3.13.2 Member Function Documentation

#### 3.13.2.1 LIMINE\_PTR() [1/2]

```
limine_framebuffer::LIMINE_PTR (
    void * )
```

#### 3.13.2.2 LIMINE\_PTR() [2/2]

```
limine_framebuffer::LIMINE_PTR (
    void * )
```

### 3.13.3 Field Documentation

#### 3.13.3.1 blue\_mask\_shift

```
uint8_t limine_framebuffer::blue_mask_shift
```

Definition at line 112 of file [limine.h](#).

#### 3.13.3.2 blue\_mask\_size

```
uint8_t limine_framebuffer::blue_mask_size
```

Definition at line 111 of file [limine.h](#).

#### 3.13.3.3 bpp

```
uint16_t limine_framebuffer::bpp
```

Definition at line 105 of file [limine.h](#).

### 3.13.3.4 edid\_size

```
uint64_t limine_framebuffer::edid_size
```

Definition at line 114 of file [limine.h](#).

### 3.13.3.5 green\_mask\_shift

```
uint8_t limine_framebuffer::green_mask_shift
```

Definition at line 110 of file [limine.h](#).

### 3.13.3.6 green\_mask\_size

```
uint8_t limine_framebuffer::green_mask_size
```

Definition at line 109 of file [limine.h](#).

### 3.13.3.7 height

```
uint64_t limine_framebuffer::height
```

Definition at line 103 of file [limine.h](#).

### 3.13.3.8 memory\_model

```
uint8_t limine_framebuffer::memory_model
```

Definition at line 106 of file [limine.h](#).

### 3.13.3.9 pitch

```
uint64_t limine_framebuffer::pitch
```

Definition at line 104 of file [limine.h](#).

### 3.13.3.10 red\_mask\_shift

```
uint8_t limine_framebuffer::red_mask_shift
```

Definition at line 108 of file [limine.h](#).

### 3.13.3.11 red\_mask\_size

```
uint8_t limine_framebuffer::red_mask_size
```

Definition at line 107 of file [limine.h](#).

### 3.13.3.12 unused

```
uint8_t limine_framebuffer::unused[7]
```

Definition at line 113 of file [limine.h](#).

### 3.13.3.13 width

```
uint64_t limine_framebuffer::width
```

Definition at line 102 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- src/include/[limine.h](#)

## 3.14 limine\_framebuffer\_request Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- [LIMINE\\_PTR](#) ([struct limine\\_framebuffer\\_response](#) \*) response

### Data Fields

- [uint64\\_t id](#) [4]
- [uint64\\_t revision](#)

### 3.14.1 Detailed Description

Definition at line 124 of file [limine.h](#).

### 3.14.2 Member Function Documentation

#### 3.14.2.1 LIMINE\_PTR()

```
limine_framebuffer_request::LIMINE_PTR (
    struct limine_framebuffer_response * )
```

### 3.14.3 Field Documentation

#### 3.14.3.1 id

```
uint64_t limine_framebuffer_request::id[4]
```

Definition at line 125 of file [limine.h](#).

#### 3.14.3.2 revision

```
uint64_t limine_framebuffer_request::revision
```

Definition at line 126 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- [src/include/limine.h](#)

## 3.15 limine\_framebuffer\_response Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- [LIMINE\\_PTR \(struct limine\\_framebuffer \\*\\*\) framebuffers](#)

## Data Fields

- `uint64_t revision`
- `uint64_t framebuffer_count`

### 3.15.1 Detailed Description

Definition at line 118 of file [limine.h](#).

### 3.15.2 Member Function Documentation

#### 3.15.2.1 LIMINE\_PTR()

```
limine_framebuffer_response::LIMINE_PTR (
    struct limine_framebuffer ** )
```

### 3.15.3 Field Documentation

#### 3.15.3.1 framebuffer\_count

```
uint64_t limine_framebuffer_response::framebuffer_count
```

Definition at line 120 of file [limine.h](#).

#### 3.15.3.2 revision

```
uint64_t limine_framebuffer_response::revision
```

Definition at line 119 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- `src/include/limine.h`

## 3.16 limine\_hhdm\_request Struct Reference

```
#include <limine.h>
```

## Public Member Functions

- LIMINE\_PTR (struct [limine\\_hhdm\\_response](#) \*) response

## Data Fields

- uint64\_t id [4]
- uint64\_t revision

### 3.16.1 Detailed Description

Definition at line [88](#) of file [limine.h](#).

### 3.16.2 Member Function Documentation

#### 3.16.2.1 LIMINE\_PTR()

```
limine_hhdm_request::LIMINE_PTR (
    struct limine\_hhdm\_response * )
```

### 3.16.3 Field Documentation

#### 3.16.3.1 id

```
uint64_t limine_hhdm_request::id[4]
```

Definition at line [89](#) of file [limine.h](#).

#### 3.16.3.2 revision

```
uint64_t limine_hhdm_request::revision
```

Definition at line [90](#) of file [limine.h](#).

The documentation for this struct was generated from the following file:

- src/include/[limine.h](#)

## 3.17 limine\_kernel\_address\_request Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- `LIMINE_PTR (struct limine_kernel_address_response *) response`

### Data Fields

- `uint64_t id [4]`
- `uint64_t revision`

#### 3.17.1 Detailed Description

Definition at line 395 of file [limine.h](#).

#### 3.17.2 Member Function Documentation

##### 3.17.2.1 LIMINE\_PTR()

```
limine_kernel_address_request::LIMINE_PTR (
    struct limine_kernel_address_response * )
```

#### 3.17.3 Field Documentation

##### 3.17.3.1 id

```
uint64_t limine_kernel_address_request::id[4]
```

Definition at line 396 of file [limine.h](#).

##### 3.17.3.2 revision

```
uint64_t limine_kernel_address_request::revision
```

Definition at line 397 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- `src/include/limine.h`

## 3.18 limine\_kernel\_file\_request Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- `LIMINE_PTR (struct limine_kernel_file_response *) response`

### Data Fields

- `uint64_t id [4]`
- `uint64_t revision`

#### 3.18.1 Detailed Description

Definition at line 302 of file [limine.h](#).

#### 3.18.2 Member Function Documentation

##### 3.18.2.1 LIMINE\_PTR()

```
limine_kernel_file_request::LIMINE_PTR (
```

```
    struct limine_kernel_file_response * )
```

#### 3.18.3 Field Documentation

##### 3.18.3.1 id

```
uint64_t limine_kernel_file_request::id[4]
```

Definition at line 303 of file [limine.h](#).

##### 3.18.3.2 revision

```
uint64_t limine_kernel_file_request::revision
```

Definition at line 304 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- `src/include/limine.h`

## 3.19 limine\_kernel\_file\_response Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- [LIMINE\\_PTR](#) (struct [limine\\_file](#) \*) kernel\_file

### Data Fields

- `uint64_t revision`

#### 3.19.1 Detailed Description

Definition at line [297](#) of file [limine.h](#).

#### 3.19.2 Member Function Documentation

##### 3.19.2.1 LIMINE\_PTR()

```
limine_kernel_file_response::LIMINE_PTR (
    struct limine\_file * )
```

#### 3.19.3 Field Documentation

##### 3.19.3.1 revision

```
uint64_t limine_kernel_file_response::revision
```

Definition at line [298](#) of file [limine.h](#).

The documentation for this struct was generated from the following file:

- `src/include/limine.h`

## 3.20 limine\_memmap\_request Struct Reference

```
#include <limine.h>
```

## Public Member Functions

- LIMINE\_PTR (struct [limine\\_memmap\\_response](#) \*) response

## Data Fields

- uint64\_t [id](#) [4]
- uint64\_t [revision](#)

### 3.20.1 Detailed Description

Definition at line [270](#) of file [limine.h](#).

### 3.20.2 Member Function Documentation

#### 3.20.2.1 LIMINE\_PTR()

```
limine_memmap_request::LIMINE_PTR (
    struct limine\_memmap\_response * )
```

### 3.20.3 Field Documentation

#### 3.20.3.1 id

```
uint64_t limine_memmap_request::id[4]
```

Definition at line [271](#) of file [limine.h](#).

#### 3.20.3.2 revision

```
uint64_t limine_memmap_request::revision
```

Definition at line [272](#) of file [limine.h](#).

The documentation for this struct was generated from the following file:

- src/include/[limine.h](#)

## 3.21 limine\_memmap\_response Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- `LIMINE_PTR (struct limine_memmap_entry **) entries`

### Data Fields

- `uint64_t revision`
- `uint64_t entry_count`

#### 3.21.1 Detailed Description

Definition at line 264 of file [limine.h](#).

#### 3.21.2 Member Function Documentation

##### 3.21.2.1 LIMINE\_PTR()

```
limine_memmap_response::LIMINE_PTR (
    struct limine_memmap_entry ** )
```

#### 3.21.3 Field Documentation

##### 3.21.3.1 entry\_count

```
uint64_t limine_memmap_response::entry_count
```

Definition at line 266 of file [limine.h](#).

##### 3.21.3.2 revision

```
uint64_t limine_memmap_response::revision
```

Definition at line 265 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- `src/include/limine.h`

## 3.22 limine\_module\_request Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- `LIMINE_PTR (struct limine_module_response *) response`

### Data Fields

- `uint64_t id [4]`
- `uint64_t revision`

#### 3.22.1 Detailed Description

Definition at line 318 of file [limine.h](#).

#### 3.22.2 Member Function Documentation

##### 3.22.2.1 LIMINE\_PTR()

```
limine_module_request::LIMINE_PTR (
    struct limine_module_response * )
```

#### 3.22.3 Field Documentation

##### 3.22.3.1 id

```
uint64_t limine_module_request::id[4]
```

Definition at line 319 of file [limine.h](#).

##### 3.22.3.2 revision

```
uint64_t limine_module_request::revision
```

Definition at line 320 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- `src/include/limine.h`

## 3.23 limine\_module\_response Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- [LIMINE\\_PTR](#) (struct [limine\\_file](#) \*\*) `modules`

### Data Fields

- `uint64_t revision`
- `uint64_t module_count`

#### 3.23.1 Detailed Description

Definition at line [312](#) of file [limine.h](#).

#### 3.23.2 Member Function Documentation

##### 3.23.2.1 LIMINE\_PTR()

```
limine_module_response::LIMINE_PTR (           struct limine_file ** )
```

#### 3.23.3 Field Documentation

##### 3.23.3.1 module\_count

```
uint64_t limine_module_response::module_count
```

Definition at line [314](#) of file [limine.h](#).

##### 3.23.3.2 revision

```
uint64_t limine_module_response::revision
```

Definition at line [313](#) of file [limine.h](#).

The documentation for this struct was generated from the following file:

- `src/include/limine.h`

## 3.24 limine\_rsdp\_request Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- `LIMINE_PTR (struct limine_rsdp_response *) response`

### Data Fields

- `uint64_t id [4]`
- `uint64_t revision`

#### 3.24.1 Detailed Description

Definition at line 333 of file [limine.h](#).

#### 3.24.2 Member Function Documentation

##### 3.24.2.1 LIMINE\_PTR()

```
limine_rsdp_request::LIMINE_PTR (
    struct limine_rsdp_response * )
```

#### 3.24.3 Field Documentation

##### 3.24.3.1 id

```
uint64_t limine_rsdp_request::id[4]
```

Definition at line 334 of file [limine.h](#).

##### 3.24.3.2 revision

```
uint64_t limine_rsdp_request::revision
```

Definition at line 335 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- `src/include/limine.h`

## 3.25 limine\_rsdp\_response Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- [LIMINE\\_PTR](#) (void \*) address

### Data Fields

- uint64\_t [revision](#)

#### 3.25.1 Detailed Description

Definition at line [328](#) of file [limine.h](#).

#### 3.25.2 Member Function Documentation

##### 3.25.2.1 LIMINE\_PTR()

```
limine_rsdp_response::LIMINE_PTR (
    void * )
```

#### 3.25.3 Field Documentation

##### 3.25.3.1 revision

```
uint64_t limine_rsdp_response::revision
```

Definition at line [329](#) of file [limine.h](#).

The documentation for this struct was generated from the following file:

- src/include/[limine.h](#)

## 3.26 limine\_smbios\_request Struct Reference

```
#include <limine.h>
```

## Public Member Functions

- LIMINE\_PTR (struct [limine\\_smbios\\_response](#) \*) response

## Data Fields

- uint64\_t id [4]
- uint64\_t revision

### 3.26.1 Detailed Description

Definition at line [349](#) of file [limine.h](#).

### 3.26.2 Member Function Documentation

#### 3.26.2.1 LIMINE\_PTR()

```
limine_smbios_request::LIMINE_PTR (
    struct limine\_smbios\_response * )
```

### 3.26.3 Field Documentation

#### 3.26.3.1 id

```
uint64_t limine_smbios_request::id[4]
```

Definition at line [350](#) of file [limine.h](#).

#### 3.26.3.2 revision

```
uint64_t limine_smbios_request::revision
```

Definition at line [351](#) of file [limine.h](#).

The documentation for this struct was generated from the following file:

- src/include/[limine.h](#)

## 3.27 limine\_smbios\_response Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- [LIMINE\\_PTR \(void \\*\) entry\\_32](#)
- [LIMINE\\_PTR \(void \\*\) entry\\_64](#)

### Data Fields

- `uint64_t revision`

#### 3.27.1 Detailed Description

Definition at line [343](#) of file [limine.h](#).

#### 3.27.2 Member Function Documentation

##### 3.27.2.1 LIMINE\_PTR() [1/2]

```
limine_smbios_response::LIMINE_PTR (
    void * )
```

##### 3.27.2.2 LIMINE\_PTR() [2/2]

```
limine_smbios_response::LIMINE_PTR (
    void * )
```

#### 3.27.3 Field Documentation

##### 3.27.3.1 revision

```
uint64_t limine_smbios_response::revision
```

Definition at line [344](#) of file [limine.h](#).

The documentation for this struct was generated from the following file:

- `src/include/limine.h`

## 3.28 limine\_smp\_request Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- [LIMINE\\_PTR](#) (struct limine\_smp\_response \*) response

### Data Fields

- uint64\_t [id](#) [4]
- uint64\_t [revision](#)
- uint64\_t [flags](#)

#### 3.28.1 Detailed Description

Definition at line [238](#) of file [limine.h](#).

#### 3.28.2 Member Function Documentation

##### 3.28.2.1 LIMINE\_PTR()

```
limine_smp_request::LIMINE_PTR (
    struct limine_smp_response * )
```

#### 3.28.3 Field Documentation

##### 3.28.3.1 flags

```
uint64_t limine_smp_request::flags
```

Definition at line [242](#) of file [limine.h](#).

##### 3.28.3.2 id

```
uint64_t limine_smp_request::id[4]
```

Definition at line [239](#) of file [limine.h](#).

### 3.28.3.3 revision

```
uint64_t limine_smp_request::revision
```

Definition at line 240 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- [src/include/limine.h](#)

## 3.29 limine\_stack\_size\_request Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- [LIMINE\\_PTR](#) ([struct limine\\_stack\\_size\\_response](#) \*) [response](#)

### Data Fields

- [uint64\\_t id](#) [4]
- [uint64\\_t revision](#)
- [uint64\\_t stack\\_size](#)

### 3.29.1 Detailed Description

Definition at line 72 of file [limine.h](#).

### 3.29.2 Member Function Documentation

#### 3.29.2.1 LIMINE\_PTR()

```
limine_stack_size_request::LIMINE_PTR (
    struct limine_stack_size_response * )
```

### 3.29.3 Field Documentation

### 3.29.3.1 id

```
uint64_t limine_stack_size_request::id[4]
```

Definition at line 73 of file [limine.h](#).

### 3.29.3.2 revision

```
uint64_t limine_stack_size_request::revision
```

Definition at line 74 of file [limine.h](#).

### 3.29.3.3 stack\_size

```
uint64_t limine_stack_size_request::stack_size
```

Definition at line 76 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- [src/include/limine.h](#)

## 3.30 limine\_terminal Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- [LIMINE\\_PTR](#) (struct [limine\\_framebuffer](#) \*) [framebuffer](#)

### Data Fields

- uint64\_t [columns](#)
- uint64\_t [rows](#)

### 3.30.1 Detailed Description

Definition at line 153 of file [limine.h](#).

### 3.30.2 Member Function Documentation

#### 3.30.2.1 LIMINE\_PTR()

```
limine_terminal::LIMINE_PTR (
    struct limine_FRAMEBUFFER * )
```

### 3.30.3 Field Documentation

#### 3.30.3.1 columns

```
uint64_t limine_terminal::columns
```

Definition at line 154 of file [limine.h](#).

#### 3.30.3.2 rows

```
uint64_t limine_terminal::rows
```

Definition at line 155 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- src/include/[limine.h](#)

## 3.31 limine\_terminal\_request Struct Reference

```
#include <limine.h>
```

### Public Member Functions

- [LIMINE\\_PTR \(struct limine\\_terminal\\_response \\*\) response](#)
- [LIMINE\\_PTR \(limine\\_terminal\\_callback\) callback](#)

### Data Fields

- uint64\_t [id](#) [4]
- uint64\_t [revision](#)

### 3.31.1 Detailed Description

Definition at line 166 of file [limine.h](#).

### 3.31.2 Member Function Documentation

#### 3.31.2.1 LIMINE\_PTR() [1/2]

```
limine_terminal_request::LIMINE_PTR (
    limine_terminal_callback )
```

#### 3.31.2.2 LIMINE\_PTR() [2/2]

```
limine_terminal_request::LIMINE_PTR (
    struct limine_terminal_response * )
```

### 3.31.3 Field Documentation

#### 3.31.3.1 id

```
uint64_t limine_terminal_request::id[4]
```

Definition at line 167 of file [limine.h](#).

#### 3.31.3.2 revision

```
uint64_t limine_terminal_request::revision
```

Definition at line 168 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- src/include/[limine.h](#)

## 3.32 limine\_terminal\_response Struct Reference

```
#include <limine.h>
```

## Public Member Functions

- `LIMINE_PTR` (`struct limine_terminal **`) `terminals`
- `LIMINE_PTR` (`limine_terminal_write`) `write`

## Data Fields

- `uint64_t revision`
- `uint64_t terminal_count`

### 3.32.1 Detailed Description

Definition at line 159 of file [limine.h](#).

### 3.32.2 Member Function Documentation

#### 3.32.2.1 LIMINE\_PTR() [1/2]

```
limine_terminal_response::LIMINE_PTR (
    limine_terminal_write   )
```

#### 3.32.2.2 LIMINE\_PTR() [2/2]

```
limine_terminal_response::LIMINE_PTR (
    struct limine_terminal **   )
```

### 3.32.3 Field Documentation

#### 3.32.3.1 revision

```
uint64_t limine_terminal_response::revision
```

Definition at line 160 of file [limine.h](#).

### 3.32.3.2 terminal\_count

```
uint64_t limine_terminal_response::terminal_count
```

Definition at line 161 of file [limine.h](#).

The documentation for this struct was generated from the following file:

- [src/include/limine.h](#)

## 3.33 out\_fct\_wrap\_type Struct Reference

### Data Fields

- `void(* fct )(char character, void *arg)`
- `void * arg`

### 3.33.1 Detailed Description

Definition at line 122 of file [printf.c](#).

### 3.33.2 Field Documentation

#### 3.33.2.1 arg

```
void* out_fct_wrap_type::arg
```

Definition at line 125 of file [printf.c](#).

#### 3.33.2.2 fct

```
void(* out_fct_wrap_type::fct) (char character, void *arg)
```

Definition at line 124 of file [printf.c](#).

The documentation for this struct was generated from the following file:

- [src/printf.c](#)

## 3.34 ScopedLock Class Reference

```
#include <lock.hpp>
```

### Public Member Functions

- [ScopedLock \(AtomicLock &value\)](#)
- [~ScopedLock \(\)](#)

#### 3.34.1 Detailed Description

Definition at line [22](#) of file [lock.hpp](#).

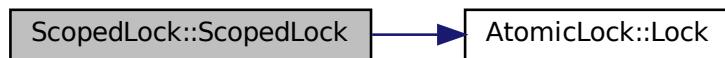
#### 3.34.2 Constructor & Destructor Documentation

##### 3.34.2.1 ScopedLock()

```
ScopedLock::ScopedLock (   
    AtomicLock & value )
```

Definition at line [39](#) of file [lock.cpp](#).

Here is the call graph for this function:



### 3.34.2.2 ~ScopedLock()

```
ScopedLock::~ScopedLock ( )
```

Definition at line 44 of file [lock.cpp](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- src/include/[lock.hpp](#)
- src/[lock.cpp](#)

## 3.35 term\_context Struct Reference

```
#include <term.h>
```

### Data Fields

- size\_t [tab\\_size](#)
- bool [autoflush](#)
- bool [scroll\\_enabled](#)
- bool [control\\_sequence](#)
- bool [csi](#)
- bool [escape](#)
- bool [rrr](#)
- bool [discard\\_next](#)
- bool [bold](#)
- bool [reverse\\_video](#)
- bool [dec\\_private](#)
- bool [insert\\_mode](#)
- uint8\_t [g\\_select](#)
- uint8\_t [charsets](#) [2]
- size\_t [current\\_charset](#)
- size\_t [escape\\_offset](#)
- size\_t [esc\\_values\\_i](#)
- size\_t [saved\\_cursor\\_x](#)
- size\_t [saved\\_cursor\\_y](#)
- size\_t [current\\_primary](#)
- size\_t [scroll\\_top\\_margin](#)
- size\_t [scroll\\_bottom\\_margin](#)

- `uint32_t esc_values [TERM_MAX_ESC_VALUES]`
- `bool saved_state_bold`
- `bool saved_state_reverse_video`
- `size_t saved_state_current_charset`
- `size_t saved_state_current_primary`
- `size_t rows`
- `size_t cols`
- `bool in_bootloader`
- `void(* raw_putchar )(struct term_context *, uint8_t c)`
- `void(* clear )(struct term_context *, bool move)`
- `void(* enable_cursor )(struct term_context *)`
- `bool(* disable_cursor )(struct term_context *)`
- `void(* set_cursor_pos )(struct term_context *, size_t x, size_t y)`
- `void(* get_cursor_pos )(struct term_context *, size_t *x, size_t *y)`
- `void(* set_text_fg )(struct term_context *, size_t fg)`
- `void(* set_text_bg )(struct term_context *, size_t bg)`
- `void(* set_text_fg_bright )(struct term_context *, size_t fg)`
- `void(* set_text_bg_bright )(struct term_context *, size_t bg)`
- `void(* set_text_fg_rgb )(struct term_context *, uint32_t fg)`
- `void(* set_text_bg_rgb )(struct term_context *, uint32_t bg)`
- `void(* set_text_fg_default )(struct term_context *)`
- `void(* set_text_bg_default )(struct term_context *)`
- `void(* move_character )(struct term_context *, size_t new_x, size_t new_y, size_t old_x, size_t old_y)`
- `void(* scroll )(struct term_context *)`
- `void(* revscroll )(struct term_context *)`
- `void(* swap_palette )(struct term_context *)`
- `void(* save_state )(struct term_context *)`
- `void(* restore_state )(struct term_context *)`
- `void(* double_buffer_flush )(struct term_context *)`
- `void(* full_refresh )(struct term_context *)`
- `void(*_deinit )(struct term_context *, void(*)(void *, size_t))`
- `void(* callback )(struct term_context *, uint64_t, uint64_t, uint64_t, uint64_t)`

### 3.35.1 Detailed Description

Definition at line 19 of file [term.h](#).

### 3.35.2 Field Documentation

#### 3.35.2.1 autoflush

```
bool term_context::autoflush
```

Definition at line 24 of file [term.h](#).

### 3.35.2.2 bold

```
bool term_context::bold
```

Definition at line [31](#) of file [term.h](#).

### 3.35.2.3 callback

```
void(* term_context::callback) (struct term_context *, uint64_t, uint64_t, uint64_t, uint64_t)
```

Definition at line [82](#) of file [term.h](#).

### 3.35.2.4 charsets

```
uint8_t term_context::charsets[2]
```

Definition at line [36](#) of file [term.h](#).

### 3.35.2.5 clear

```
void(* term_context::clear) (struct term_context *, bool move)
```

Definition at line [57](#) of file [term.h](#).

### 3.35.2.6 cols

```
size_t term_context::cols
```

Definition at line [53](#) of file [term.h](#).

### 3.35.2.7 control\_sequence

```
bool term_context::control_sequence
```

Definition at line [26](#) of file [term.h](#).

### 3.35.2.8 `csi`

```
bool term_context::csi
```

Definition at line [27](#) of file [term.h](#).

### 3.35.2.9 `current_charset`

```
size_t term_context::current_charset
```

Definition at line [37](#) of file [term.h](#).

### 3.35.2.10 `current_primary`

```
size_t term_context::current_primary
```

Definition at line [42](#) of file [term.h](#).

### 3.35.2.11 `dec_private`

```
bool term_context::dec_private
```

Definition at line [33](#) of file [term.h](#).

### 3.35.2.12 `deinit`

```
void(* term_context::deinit) (struct term_context *, void(*)(void *, size_t))
```

Definition at line [78](#) of file [term.h](#).

### 3.35.2.13 `disable_cursor`

```
bool(* term_context::disable_cursor) (struct term_context *)
```

Definition at line [59](#) of file [term.h](#).

### 3.35.2.14 discard\_next

```
bool term_context::discard_next
```

Definition at line 30 of file [term.h](#).

### 3.35.2.15 double\_buffer\_flush

```
void(* term_context::double_buffer_flush) (struct term\_context *)
```

Definition at line 76 of file [term.h](#).

### 3.35.2.16 enable\_cursor

```
void(* term_context::enable_cursor) (struct term\_context *)
```

Definition at line 58 of file [term.h](#).

### 3.35.2.17 esc\_values

```
uint32_t term_context::esc_values[TERM\_MAX\_ESC\_VALUES]
```

Definition at line 45 of file [term.h](#).

### 3.35.2.18 esc\_values\_i

```
size_t term_context::esc_values_i
```

Definition at line 39 of file [term.h](#).

### 3.35.2.19 escape

```
bool term_context::escape
```

Definition at line 28 of file [term.h](#).

### 3.35.2.20 escape\_offset

```
size_t term_context::escape_offset
```

Definition at line [38](#) of file [term.h](#).

### 3.35.2.21 full\_refresh

```
void(* term_context::full_refresh) (struct term_context *)
```

Definition at line [77](#) of file [term.h](#).

### 3.35.2.22 g\_select

```
uint8_t term_context::g_select
```

Definition at line [35](#) of file [term.h](#).

### 3.35.2.23 get\_cursor\_pos

```
void(* term_context::get_cursor_pos) (struct term_context *, size_t *x, size_t *y)
```

Definition at line [61](#) of file [term.h](#).

### 3.35.2.24 in\_bootloader

```
bool term_context::in_bootloader
```

Definition at line [54](#) of file [term.h](#).

### 3.35.2.25 insert\_mode

```
bool term_context::insert_mode
```

Definition at line [34](#) of file [term.h](#).

### 3.35.2.26 move\_character

```
void(* term_context::move_character) (struct term_context *, size_t new_x, size_t new_y, size_t old_x, size_t old_y)
```

Definition at line [70](#) of file [term.h](#).

### 3.35.2.27 raw\_putchar

```
void(* term_context::raw_putchar) (struct term_context *, uint8_t c)
```

Definition at line [56](#) of file [term.h](#).

### 3.35.2.28 restore\_state

```
void(* term_context::restore_state) (struct term_context *)
```

Definition at line [75](#) of file [term.h](#).

### 3.35.2.29 reverse\_video

```
bool term_context::reverse_video
```

Definition at line [32](#) of file [term.h](#).

### 3.35.2.30 revscroll

```
void(* term_context::revscroll) (struct term_context *)
```

Definition at line [72](#) of file [term.h](#).

### 3.35.2.31 rows

```
size_t term_context::rows
```

Definition at line [53](#) of file [term.h](#).

### 3.35.2.32 `rrr`

```
bool term_context::rrr
```

Definition at line [29](#) of file [term.h](#).

### 3.35.2.33 `save_state`

```
void(* term_context::save_state) (struct term_context *)
```

Definition at line [74](#) of file [term.h](#).

### 3.35.2.34 `saved_cursor_x`

```
size_t term_context::saved_cursor_x
```

Definition at line [40](#) of file [term.h](#).

### 3.35.2.35 `saved_cursor_y`

```
size_t term_context::saved_cursor_y
```

Definition at line [41](#) of file [term.h](#).

### 3.35.2.36 `saved_state_bold`

```
bool term_context::saved_state_bold
```

Definition at line [46](#) of file [term.h](#).

### 3.35.2.37 `saved_state_current_charset`

```
size_t term_context::saved_state_current_charset
```

Definition at line [48](#) of file [term.h](#).

### 3.35.2.38 saved\_state\_current\_primary

```
size_t term_context::saved_state_current_primary
```

Definition at line [49](#) of file [term.h](#).

### 3.35.2.39 saved\_state\_reverse\_video

```
bool term_context::saved_state_reverse_video
```

Definition at line [47](#) of file [term.h](#).

### 3.35.2.40 scroll

```
void(* term_context::scroll) (struct term_context *)
```

Definition at line [71](#) of file [term.h](#).

### 3.35.2.41 scroll\_bottom\_margin

```
size_t term_context::scroll_bottom_margin
```

Definition at line [44](#) of file [term.h](#).

### 3.35.2.42 scroll\_enabled

```
bool term_context::scroll_enabled
```

Definition at line [25](#) of file [term.h](#).

### 3.35.2.43 scroll\_top\_margin

```
size_t term_context::scroll_top_margin
```

Definition at line [43](#) of file [term.h](#).

### 3.35.2.44 `set_cursor_pos`

```
void(* term_context::set_cursor_pos) (struct term_context *, size_t x, size_t y)
```

Definition at line [60](#) of file `term.h`.

### 3.35.2.45 `set_text_bg`

```
void(* term_context::set_text_bg) (struct term_context *, size_t bg)
```

Definition at line [63](#) of file `term.h`.

### 3.35.2.46 `set_text_bg_bright`

```
void(* term_context::set_text_bg_bright) (struct term_context *, size_t bg)
```

Definition at line [65](#) of file `term.h`.

### 3.35.2.47 `set_text_bg_default`

```
void(* term_context::set_text_bg_default) (struct term_context *)
```

Definition at line [69](#) of file `term.h`.

### 3.35.2.48 `set_text_bg_rgb`

```
void(* term_context::set_text_bg_rgb) (struct term_context *, uint32_t bg)
```

Definition at line [67](#) of file `term.h`.

### 3.35.2.49 `set_text_fg`

```
void(* term_context::set_text_fg) (struct term_context *, size_t fg)
```

Definition at line [62](#) of file `term.h`.

### 3.35.2.50 set\_text\_fg\_bright

```
void(* term_context::set_text_fg_bright) (struct term\_context *, size_t fg)
```

Definition at line [64](#) of file [term.h](#).

### 3.35.2.51 set\_text\_fg\_default

```
void(* term_context::set_text_fg_default) (struct term\_context *)
```

Definition at line [68](#) of file [term.h](#).

### 3.35.2.52 set\_text\_fg\_rgb

```
void(* term_context::set_text_fg_rgb) (struct term\_context *, uint32_t fg)
```

Definition at line [66](#) of file [term.h](#).

### 3.35.2.53 swap\_palette

```
void(* term_context::swap_palette) (struct term\_context *)
```

Definition at line [73](#) of file [term.h](#).

### 3.35.2.54 tab\_size

```
size_t term_context::tab_size
```

Definition at line [23](#) of file [term.h](#).

The documentation for this struct was generated from the following file:

- src/include/terminal/[term.h](#)

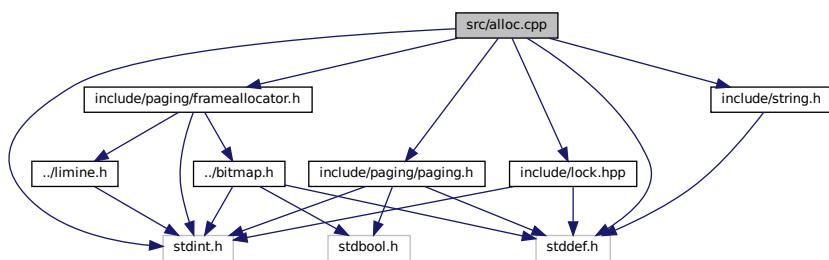


# Chapter 4

## File Documentation

### 4.1 src/alloc.cpp File Reference

```
#include <stdint.h>
#include <stddef.h>
#include "include/string.h"
#include "include/paging/frameallocator.h"
#include "include/paging/paging.h"
#include "include/lock.hpp"
Include dependency graph for alloc.cpp:
```



### Functions

- int [liballoc\\_lock \(\)](#)
- int [liballoc\\_unlock \(\)](#)
- void \* [liballoc\\_alloc \(size\\_t pages\)](#)
- int [liballoc\\_free \(void \\*ptr, size\\_t pages\)](#)

### Variables

- [AtomicLock liballocLock](#)

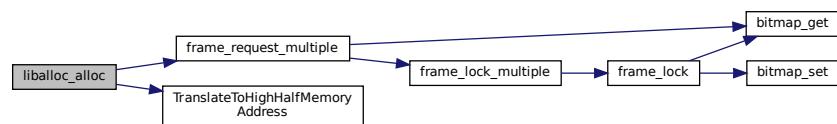
## 4.1.1 Function Documentation

### 4.1.1.1 liballoc\_alloc()

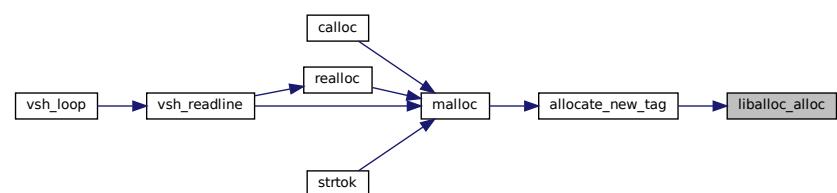
```
void* liballoc_alloc (
    size_t pages )
```

Definition at line 26 of file [alloc.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

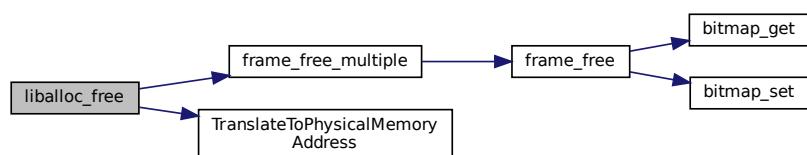


### 4.1.1.2 liballoc\_free()

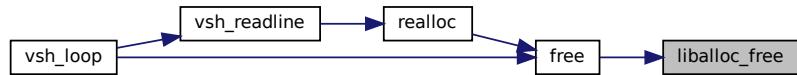
```
int liballoc_free (
    void * ptr,
    size_t pages )
```

Definition at line 42 of file [alloc.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.1.1.3 liballoc\_lock()

```
int liballoc_lock ( )
```

This function is supposed to lock the memory data structures. It could be as simple as disabling interrupts or acquiring a spinlock. It's up to you to decide.

##### Returns

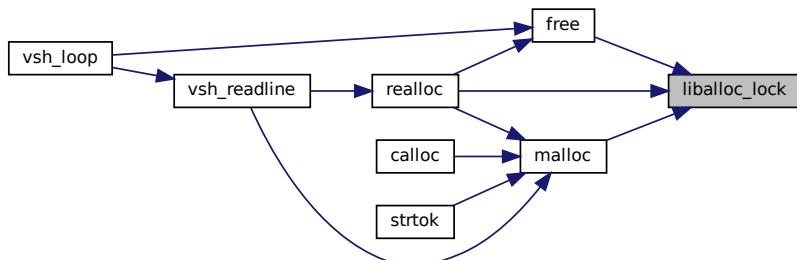
0 if the lock was acquired successfully. Anything else is failure.

Definition at line 10 of file [alloc.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.1.1.4 liballoc\_unlock()

```
int liballoc_unlock ( )
```

This function unlocks what was previously locked by the liballoc\_lock function. If it disabled interrupts, it enables interrupts. If it had acquired a spinlock, it releases the spinlock. etc.

##### Returns

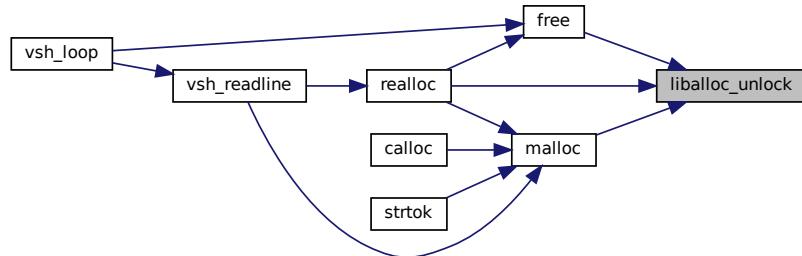
0 if the lock was successfully released.

Definition at line 18 of file [alloc.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.1.2 Variable Documentation

##### 4.1.2.1 liballocLock

```
AtomicLock liballocLock
```

Definition at line 8 of file [alloc.cpp](#).

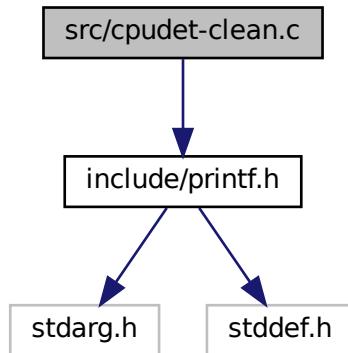
## 4.2 alloc.cpp

```
00001 #include <stdint.h>
00002 #include <stddef.h>
00003 #include "include/string.h"
00004 #include "include/paging/frameallocator.h"
00005 #include "include/paging/paging.h"
00006 #include "include/lock.hpp"
00007
00008 AtomicLock liballocLock;
00009
00010 extern "C" int liballoc_lock()
00011 {
00012     liballocLock.Lock();
00014
00015     return 0;
00016 }
00017
00018 extern "C" int liballoc_unlock()
00019 {
00020
00021     liballocLock.Unlock();
00022
00023     return 0;
00024 }
00025
00026 extern "C" void *liballoc_alloc(size_t pages)
00027 {
00028
00029     void *ptr = frame_request_multiple(pages);
00030
00031     if (ptr == NULL)
00032     {
00033
00034         return NULL;
00035     }
00036
00037     void *realPtr = (void *)TranslateToHighHalfMemoryAddress((uint64_t)ptr);
00038
00039     return realPtr;
00040 }
00041
00042 extern "C" int liballoc_free(void *ptr, size_t pages)
00043 {
00044
00045     void *realPtr = (void *)TranslateToPhysicalMemoryAddress((uint64_t)ptr);
00046
00047     frame_free_multiple(realPtr, pages);
00048
00049     return 0;
00050 }
```

## 4.3 src/cpudet-clean.c File Reference

```
#include "include/printf.h"
```

Include dependency graph for cpudet-clean.c:



## Macros

- `#define cpuid(in, a, b, c, d)`

## Functions

- `int do_intel (void)`
- `int do_amd (void)`
- `void printregs (int eax, int ebx, int ecx, int edx)`
- `void detect_cpu (void)`

## Variables

- `char * Intel []`
- `char * Intel_Other []`

### 4.3.1 Macro Definition Documentation

#### 4.3.1.1 cpuid

```
#define cpuid(
    in,
    a,
    b,
    c,
    d )
```

##### Value:

```
_asm_ ("cpuid"
: "=a"(a), "=b"(b), "=c"(c), "=d"(d) \
: "a"(in)); \
```

Definition at line 37 of file [cpudet-clean.c](#).

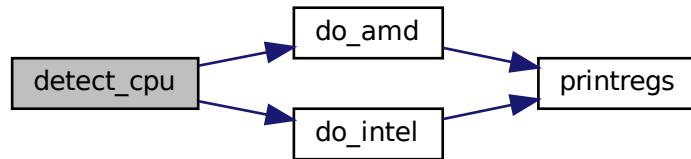
### 4.3.2 Function Documentation

#### 4.3.2.1 detect\_cpu()

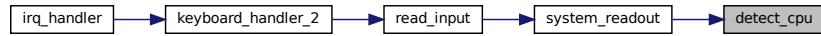
```
void detect_cpu (
    void )
```

Definition at line 42 of file [cpudet-clean.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.3.2.2 do\_amd()

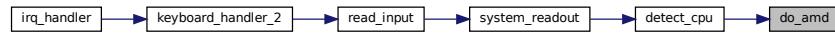
```
int do_amd (
    void )
```

Definition at line 313 of file [cpudet-clean.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.3.2.3 do\_intel()

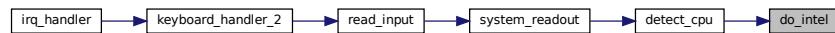
```
int do_intel (
    void )
```

Definition at line 116 of file [cpudet-clean.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

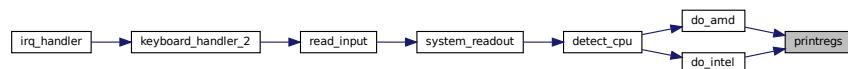


#### 4.3.2.4 printregs()

```
void printregs (
    int eax,
    int ebx,
    int ecx,
    int edx )
```

Definition at line 297 of file [cpudet-clean.c](#).

Here is the caller graph for this function:



### 4.3.3 Variable Documentation

#### 4.3.3.1 Intel

```
char* Intel[ ]
```

**Initial value:**

```
= {
    "Brand ID Not Supported.",
    "Intel(R) Celeron(R) processor",
    "Intel(R) Pentium(R) III processor",
    "Intel(R) Pentium(R) III Xeon(R) processor",
    "Intel(R) Pentium(R) III processor",
    "Reserved",
    "Mobile Intel(R) Pentium(R) III processor-M",
    "Mobile Intel(R) Celeron(R) processor",
    "Intel(R) Pentium(R) 4 processor",
    "Intel(R) Pentium(R) 4 processor",
    "Intel(R) Celeron(R) processor",
    "Intel(R) Xeon(R) Processor",
    "Intel(R) Xeon(R) processor MP",
    "Reserved",
    "Mobile Intel(R) Pentium(R) 4 processor-M",
    "Mobile Intel(R) Pentium(R) Celeron(R) processor",
    "Reserved",
    "Mobile Genuine Intel(R) processor",
    "Intel(R) Celeron(R) M processor",
    "Mobile Intel(R) Celeron(R) processor",
    "Intel(R) Celeron(R) processor",
    "Mobile Genuine Intel(R) processor",
    "Intel(R) Pentium(R) M processor",
    "Mobile Intel(R) Celeron(R) processor"
}
```

Definition at line 62 of file [cpudet-clean.c](#).

#### 4.3.3.2 Intel\_Other

```
char* Intel_Other[ ]
```

**Initial value:**

```
= {
    "Reserved",
    "Reserved",
    "Reserved",
    "Reserved",
    "Intel(R) Celeron(R) processor",
    "Reserved",
    "Intel(R) Xeon(R) processor MP",
    "Reserved",
    "Reserved",
    "Reserved",
    "Intel(R) Xeon(R) processor",
    "Reserved",
    "Reserved",
    "Reserved",
    "Reserved",
    "Reserved",
    "Reserved",
    "Reserved",
    "Reserved",
    "Reserved",
    "Reserved"
}
```

Definition at line 89 of file [cpudet-clean.c](#).

## 4.4 cpudet-clean.c

```

00001 /*
00002 * Copyright (c) 2006-2007 - http://brynet.biz.tm - <brynet@gmail.com>
00003 * All rights reserved.
00004 *
00005 * Redistribution and use in source and binary forms, with or without
00006 * modification, are permitted provided that the following conditions
00007 * are met:
00008 * 1. Redistributions of source code must retain the above copyright
00009 * notice, this list of conditions and the following disclaimer.
00010 * 2. Redistributions in binary form must reproduce the above copyright
00011 * notice, this list of conditions and the following disclaimer in the
00012 * documentation and/or other materials provided with the distribution.
00013 * 3. The name of the author may not be used to endorse or promote products
00014 * derived from this software without specific prior written permission.
00015 *
00016 * THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
00017 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
00018 * AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
00019 * THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
00020 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00021 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
00022 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
00023 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
00024 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
00025 * ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00026 */
00027
00028 /* You need to include a file with fairly(ish) compliant printf prototype, Decimal and String support
   like %s and %d and this is truely all you need! */
00029 // #include <stdio.h> /* for printf(); */
00030 #include "include/printf.h"
00031
00032 /* Required Declarations */
00033 int do_intel(void);
00034 int do_amd(void);
00035 void printregs(int eax, int ebx, int ecx, int edx);
00036
00037 #define cpuid(in, a, b, c, d) __asm__("cpuid"
00038 : "=a"(a), "=b"(b), "=c"(c), "=d"(d) \
00039 : "a"(in));
00040
00041 /* Simply call this function detect_cpu(); */
00042 void detect_cpu(void)
00043 { /* or main() if your trying to port this as an independant application */
00044     unsigned long ebx, unused;
00045     cpuid(0, unused, ebx, unused, unused);
00046     switch (ebx)
00047     {
00048         case 0x756e6547: /* Intel Magic Code */
00049             do_intel();
00050             break;
00051         case 0x68747541: /* AMD Magic Code */
00052             do_amd();
00053             break;
00054         default:
00055             printf("Unknown x86 CPU Detected\n");
00056             break;
00057     }
00058     return;
00059 }
00060
00061 /* Intel Specific brand list */
00062 char *Intel[] =
00063 {"Brand ID Not Supported.",
00064 "Intel(R) Celeron(R) processor",
00065 "Intel(R) Pentium(R) III processor",
00066 "Intel(R) Pentium(R) III Xeon(R) processor",
00067 "Intel(R) Pentium(R) III processor",
00068 "Reserved",
00069 "Mobile Intel(R) Pentium(R) III processor-M",
00070 "Mobile Intel(R) Celeron(R) processor",
00071 "Intel(R) Pentium(R) 4 processor",
00072 "Intel(R) Pentium(R) 4 processor",
00073 "Intel(R) Celeron(R) processor",
00074 "Intel(R) Xeon(R) Processor",
00075 "Intel(R) Xeon(R) processor MP",
00076 "Reserved",
00077 "Mobile Intel(R) Pentium(R) 4 processor-M",
00078 "Mobile Intel(R) Pentium(R) Celeron(R) processor",
00079 "Reserved",
00080 "Mobile Genuine Intel(R) processor",
00081 "Intel(R) Celeron(R) M processor",
00082 "Mobile Intel(R) Celeron(R) processor",
00083 "Intel(R) Celeron(R) processor",
00084 "Mobile Geniune Intel(R) processor",

```

```
00085     "Intel(R) Pentium(R) M processor",
00086     "Mobile Intel(R) Celeron(R) processor"};
00087
00088 /* This table is for those brand strings that have two values depending on the processor signature. It
00089    should have the same number of entries as the above table. */
00090 char *Intel_Other[] = {
00091     "Reserved",
00092     "Reserved",
00093     "Reserved",
00094     "Intel(R) Celeron(R) processor",
00095     "Reserved",
00096     "Reserved",
00097     "Reserved",
00098     "Reserved",
00099     "Reserved",
00100     "Reserved",
00101     "Intel(R) Xeon(R) processor MP",
00102     "Reserved",
00103     "Reserved",
00104     "Intel(R) Xeon(R) processor",
00105     "Reserved",
00106     "Reserved",
00107     "Reserved",
00108     "Reserved",
00109     "Reserved",
00110     "Reserved",
00111     "Reserved",
00112     "Reserved",
00113     "Reserved"};
00114
00115 /* Intel-specific information */
00116 int do_intel(void)
00117 {
00118     printf("Intel Specific Features:\n");
00119     unsigned long eax, ebx, ecx, edx, max_eax, signature, unused;
00120     int model, family, type, brand, stepping, reserved;
00121     int extended_family = -1;
00122     cpuid(1, eax, ebx, unused, unused);
00123     model = (eax >> 4) & 0xf;
00124     family = (eax >> 8) & 0xf;
00125     type = (eax >> 12) & 0x3;
00126     brand = ebx & 0xff;
00127     stepping = eax & 0xf;
00128     reserved = eax >> 14;
00129     signature = eax;
00130     printf("Type %d - ", type);
00131     switch (type)
00132     {
00133         case 0:
00134             printf("Original OEM");
00135             break;
00136         case 1:
00137             printf("Overdrive");
00138             break;
00139         case 2:
00140             printf("Dual-capable");
00141             break;
00142         case 3:
00143             printf("Reserved");
00144             break;
00145     }
00146     printf("\n");
00147     printf("Family %d - ", family);
00148     switch (family)
00149     {
00150         case 3:
00151             printf("i386");
00152             break;
00153         case 4:
00154             printf("i486");
00155             break;
00156         case 5:
00157             printf("Pentium");
00158             break;
00159         case 6:
00160             printf("Pentium Pro");
00161             break;
00162         case 15:
00163             printf("Pentium 4");
00164     }
00165     printf("\n");
00166     if (family == 15)
00167     {
00168         extended_family = (eax >> 20) & 0xff;
00169         printf("Extended family %d\n", extended_family);
00170     }
```

```

00171     printf("Model %d - ", model);
00172     switch (family)
00173     {
00174         case 3:
00175             break;
00176         case 4:
00177             switch (model)
00178             {
00179                 case 0:
00180                 case 1:
00181                     printf("DX");
00182                     break;
00183                 case 2:
00184                     printf("SX");
00185                     break;
00186                 case 3:
00187                     printf("487/DX2");
00188                     break;
00189                 case 4:
00190                     printf("SL");
00191                     break;
00192                 case 5:
00193                     printf("SX2");
00194                     break;
00195                 case 7:
00196                     printf("Write-back enhanced DX2");
00197                     break;
00198                 case 8:
00199                     printf("DX4");
00200                     break;
00201             }
00202             break;
00203         case 5:
00204             switch (model)
00205             {
00206                 case 1:
00207                     printf("60/66");
00208                     break;
00209                 case 2:
00210                     printf("75-200");
00211                     break;
00212                 case 3:
00213                     printf("for 486 system");
00214                     break;
00215                 case 4:
00216                     printf("MMX");
00217                     break;
00218             }
00219             break;
00220         case 6:
00221             switch (model)
00222             {
00223                 case 1:
00224                     printf("Pentium Pro");
00225                     break;
00226                 case 3:
00227                     printf("Pentium II Model 3");
00228                     break;
00229                 case 5:
00230                     printf("Pentium II Model 5/Xeon/Celeron");
00231                     break;
00232                 case 6:
00233                     printf("Celeron");
00234                     break;
00235                 case 7:
00236                     printf("Pentium III/Pentium III Xeon - external L2 cache");
00237                     break;
00238                 case 8:
00239                     printf("Pentium III/Pentium III Xeon - internal L2 cache");
00240                     break;
00241             }
00242             break;
00243         case 15:
00244             break;
00245     }
00246     printf("\n");
00247     cpuid(0x80000000, max_eax, unused, unused, unused);
00248     /* Quok said: If the max extended eax value is high enough to support the processor brand string
00249     (values 0x80000002 to 0x80000004), then we'll use that information to return the brand
00250     information.
00251     Otherwise, we'll refer back to the brand tables above for backwards compatibility with older
00252     processors.
00253     According to the Sept. 2006 Intel Arch Software Developer's Guide, if extended eax values are
00254     supported,
00255     then all 3 values for the processor brand string are supported, but we'll test just to make sure
00256     and be safe. */
00257     if (max_eax >= 0x80000004)

```

```

00254     {
00255         printf("Brand: ");
00256         if (max_eax >= 0x80000002)
00257         {
00258             cpuid(0x80000002, eax, ebx, ecx, edx);
00259             printregs(eax, ebx, ecx, edx);
00260         }
00261         if (max_eax >= 0x80000003)
00262         {
00263             cpuid(0x80000003, eax, ebx, ecx, edx);
00264             printregs(eax, ebx, ecx, edx);
00265         }
00266         if (max_eax >= 0x80000004)
00267         {
00268             cpuid(0x80000004, eax, ebx, ecx, edx);
00269             printregs(eax, ebx, ecx, edx);
00270         }
00271         printf("\n");
00272     }
00273     else if (brand > 0)
00274     {
00275         printf("Brand %d - ", brand);
00276         if (brand < 0x18)
00277         {
00278             if (signature == 0x000006B1 || signature == 0x00000F13)
00279             {
00280                 printf("%s\n", Intel_Other[brand]);
00281             }
00282             else
00283             {
00284                 printf("%s\n", Intel[brand]);
00285             }
00286         }
00287         else
00288         {
00289             printf("Reserved\n");
00290         }
00291     }
00292     printf("Stepping: %d Reserved: %d\n", stepping, reserved);
00293     return 0;
00294 }
00295
00296 /* Print Registers */
00297 void printregs(int eax, int ebx, int ecx, int edx)
00298 {
00299     int j;
00300     char string[17];
00301     string[16] = '\0';
00302     for (j = 0; j < 4; j++)
00303     {
00304         string[j] = eax >> (8 * j);
00305         string[j + 4] = ebx >> (8 * j);
00306         string[j + 8] = ecx >> (8 * j);
00307         string[j + 12] = edx >> (8 * j);
00308     }
00309     printf("%s", string);
00310 }
00311
00312 /* AMD-specific information */
00313 int do_amd(void)
00314 {
00315     printf("AMD Specific Features:\n");
00316     unsigned long extended, eax, ebx, ecx, edx, unused;
00317     int family, model, stepping, reserved;
00318     cpuid(1, eax, unused, unused, unused);
00319     model = (eax >> 4) & 0xf;
00320     family = (eax >> 8) & 0xf;
00321     stepping = eax & 0xf;
00322     reserved = eax >> 12;
00323     printf("Family: %d Model: %d [", family, model);
00324     switch (family)
00325     {
00326     case 4:
00327         printf("486 Model %d", model);
00328         break;
00329     case 5:
00330         switch (model)
00331         {
00332             case 0:
00333             case 1:
00334             case 2:
00335             case 3:
00336             case 6:
00337             case 7:
00338                 printf("K6 Model %d", model);
00339                 break;
00340             case 8:

```

```

00341         printf("K6-2 Model 8");
00342         break;
00343     case 9:
00344         printf("K6-III Model 9");
00345         break;
00346     default:
00347         printf("K5/K6 Model %d", model);
00348         break;
00349     }
00350     break;
00351 case 6:
00352     switch (model)
00353     {
00354     case 1:
00355     case 2:
00356     case 4:
00357         printf("Athlon Model %d", model);
00358         break;
00359     case 3:
00360         printf("Duron Model 3");
00361         break;
00362     case 6:
00363         printf("Athlon MP/Mobile Athlon Model 6");
00364         break;
00365     case 7:
00366         printf("Mobile Duron Model 7");
00367         break;
00368     default:
00369         printf("Duron/Athlon Model %d", model);
00370         break;
00371     }
00372     break;
00373 }
00374 printf("]\n");
00375 cpuid(0x80000000, extended, unused, unused, unused);
00376 if (extended == 0)
00377 {
00378     return 0;
00379 }
00380 if (extended >= 0x80000002)
00381 {
00382     unsigned int j;
00383     printf("Detected Processor Name: ");
00384     for (j = 0x80000002; j <= 0x80000004; j++)
00385     {
00386         cpuid(j, eax, ebx, ecx, edx);
00387         printregs(eax, ebx, ecx, edx);
00388     }
00389     printf("\n");
00390 }
00391 if (extended >= 0x80000007)
00392 {
00393     cpuid(0x80000007, unused, unused, unused, edx);
00394     if (edx & 1)
00395     {
00396         printf("Temperature Sensing Diode Detected!\n");
00397     }
00398 }
00399 printf("Stepping: %d Reserved: %d\n", stepping, reserved);
00400 return 0;
00401 }

```

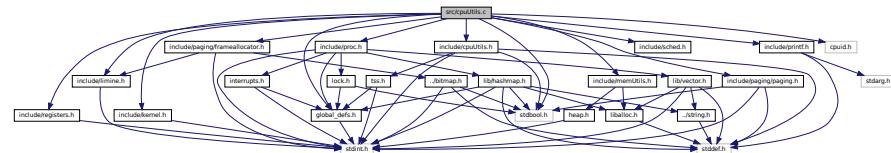
## 4.5 src/cpuUtils.c File Reference

```

#include "include/cpuUtils.h"
#include "include/printf.h"
#include "include/registers.h"
#include "include/global_defs.h"
#include "include/limine.h"
#include "include/memUtils.h"
#include "include/paging/frameallocator.h"
#include "include/paging/paging.h"
#include "include/proc.h"
#include "include/sched.h"
#include "include/kernel.h"
#include <cpuid.h>

```

```
#include <stdbool.h>
Include dependency graph for cpuUtils.c:
```



## Functions

- int `cpuid_check_sse ()`
- int `cpuid_check_xsave ()`
- int `cpuid_check_pcid ()`
- int `cpuid_check_pae ()`
- int `cpuid_check_mce ()`
- int `cpuid_check_apic ()`
- int `cpuid_check_mca ()`
- int `cpuid_check_acpi ()`
- int `cpuid_check_ds ()`
- int `cpuid_check_tm ()`
- int `cpuid_check_vmx ()`
- int `cpuid_check_htt ()`
- int `cpuid_check_fpu ()`
- int `cpuid_check_msр ()`
- int `cpuid_check_oxsave ()`
- int `cpuid_check_avx ()`
- int `cpuid_check_fxsr ()`
- int `vendor_str1 ()`
- int `vendor_str2 ()`
- int `vendor_str3 ()`
- void `halt ()`
- void `cpuid_readout ()`

*Print a message to the standard output. This is called from cpuid\_read ()*

- int `get_model (void)`

*Get the CPUID model.*

- void `get_vendor ()`

*Get vendor information from CPUID.*

- void `fpu_init ()`
- void `no_sse ()`

*Print message to indicate SSE extensions are unavailable.*

- void `no_xsave ()`

*Display message to indicate XSAVE extensions are unavailable.*

- void `check_sse ()`

*Check SSE extensions and print results if they are available.*

- void `check_xsave ()`

*Check if XSAVE is available.*

- void `check_fpu ()`

*Check if FPU is enabled or.*

- void `check_msр ()`

- void `check_vmx ()`  
*Check MSR on / off.*
- void `check_htt ()`  
*Check if HTT is enabled.*
- void `check_pcid ()`  
*Check if CPUID is available.*
- void `check_pae ()`  
*Check if PAE is installed.*
- void `check_mce ()`  
*Check if MCE is enabled.*
- void `check_apic ()`  
*Check if APIC is available.*
- void `check_mca ()`  
*Check if MCA is installed.*
- void `check_acpi ()`  
*Check if CPUID has ACPI installed.*
- void `check_ds ()`  
*Check if cpuid\_check\_ds is available.*
- void `check_tm ()`  
*Check TM and print yes / no.*

## Variables

- char `CPU_vendor [13]`
- bool `first_run = true`

### 4.5.1 Function Documentation

#### 4.5.1.1 `check_acpi()`

```
void check_acpi ( )
```

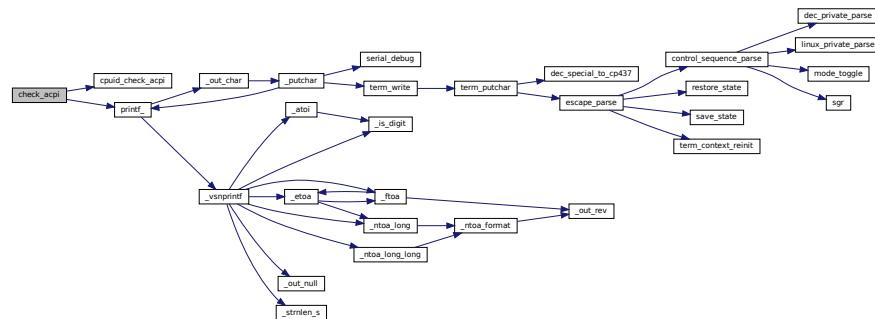
Check if CPUID has ACPI installed.

**Returns**

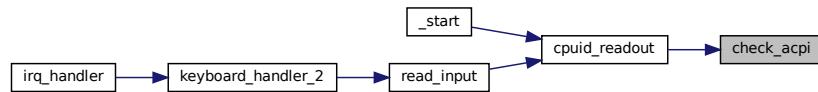
Yes if installed No

Definition at line 391 of file [cpuUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**4.5.1.2 check\_apic()**

```
void check_apic( )
```

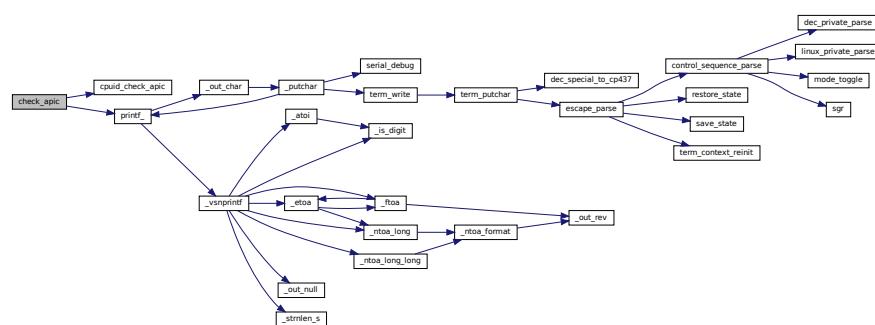
Check if APIC is available.

**Returns**

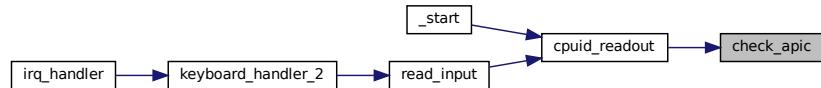
true if it is

Definition at line 349 of file [cpuUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.5.1.3 check\_ds()

```
void check_ds( )
```

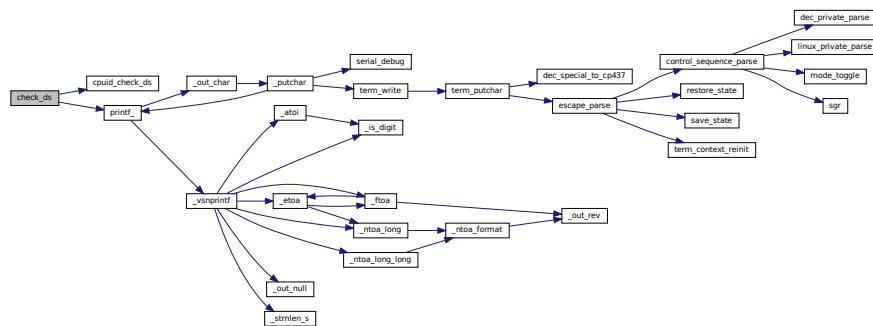
Check if cpuid\_check\_ds is available.

##### Returns

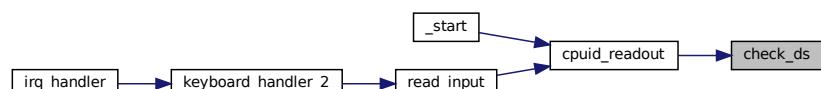
true if cpuid\_check\_ds is

Definition at line 412 of file [cpuUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



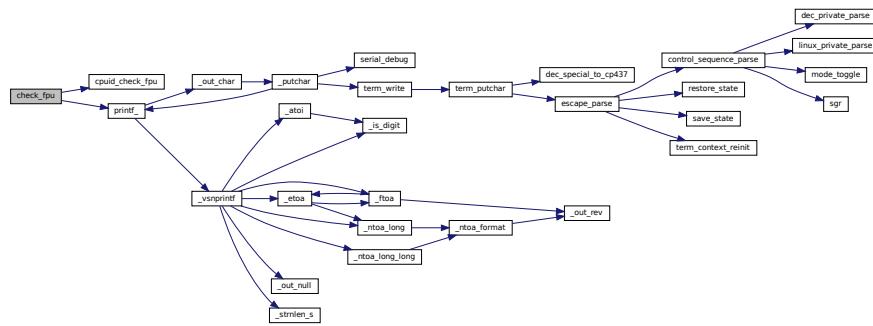
#### 4.5.1.4 check\_fpu()

```
void check_fpu ( )
```

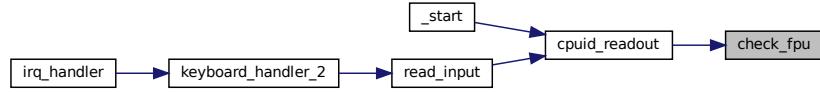
Check if FPU is enabled or.

Definition at line 213 of file [cpuUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.5.1.5 check\_htt()

```
void check_htt ( )
```

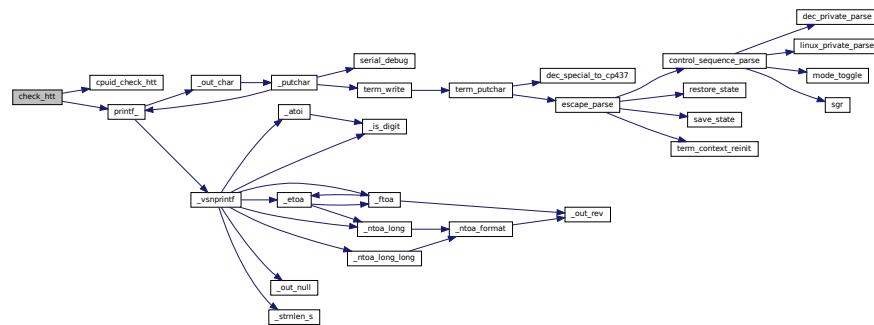
Check if HTT is enabled.

**Returns**

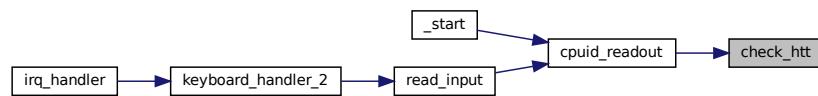
true if HTT is

Definition at line 269 of file [cpuUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**4.5.1.6 check\_mca()**

```
void check_mca ( )
```

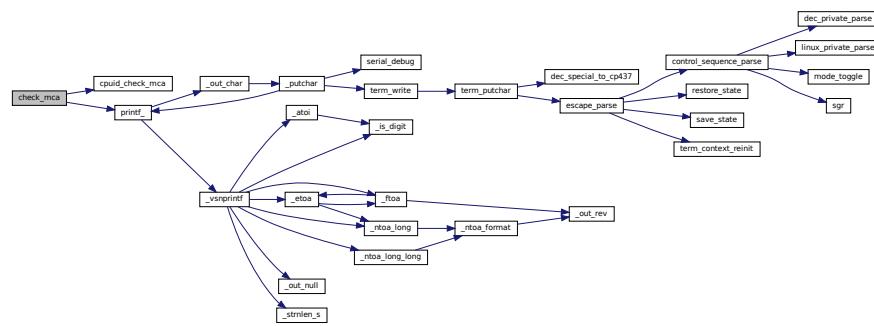
Check if MCA is installed.

**Returns**

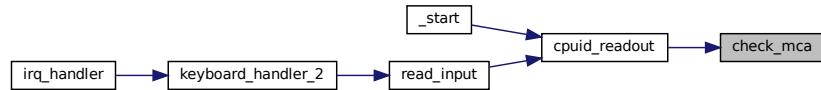
true if MCA is

Definition at line 370 of file [cpuUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.5.1.7 check\_mce()

```
void check_mce( )
```

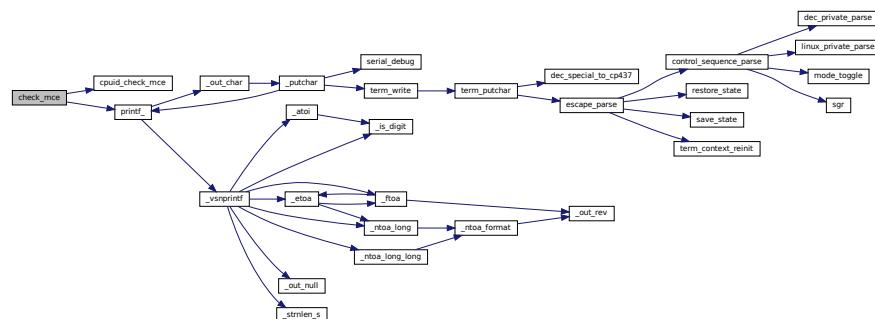
Check if MCE is enabled.

##### Returns

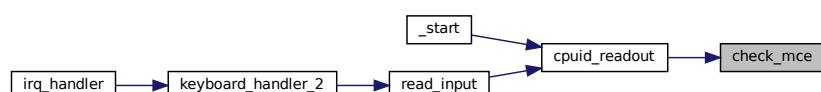
true if enabled false

Definition at line 328 of file [cpuUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.5.1.8 check\_msr()

```
void check_msr ( )
```

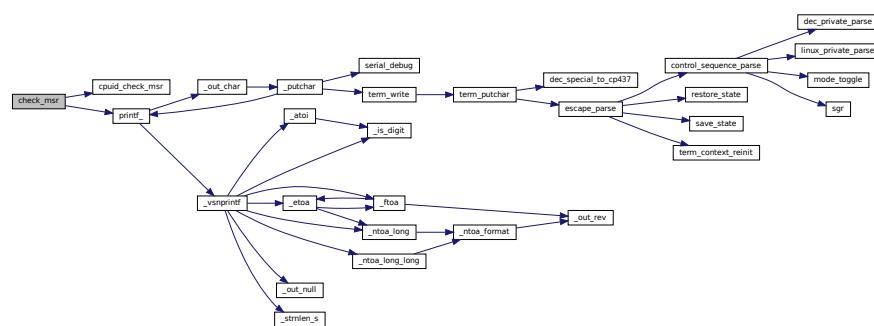
Check MSR on / off.

##### Returns

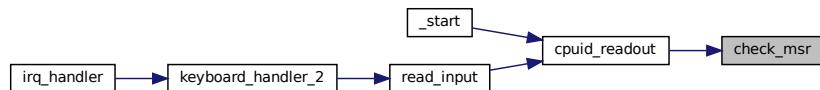
true if MSR is

Definition at line 232 of file [cpuUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.5.1.9 check\_pae()

```
void check_pae ( )
```

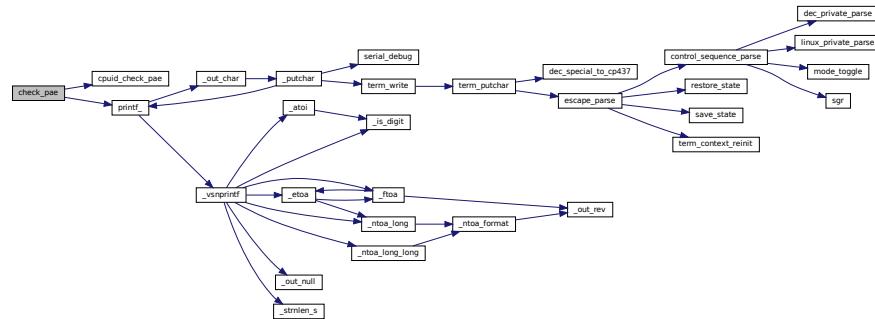
Check if PAE is installed.

**Returns**

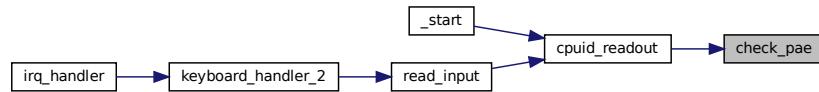
true if installed false

Definition at line 307 of file [cpuUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**4.5.1.10 check\_pcid()**

```
void check_pcid( )
```

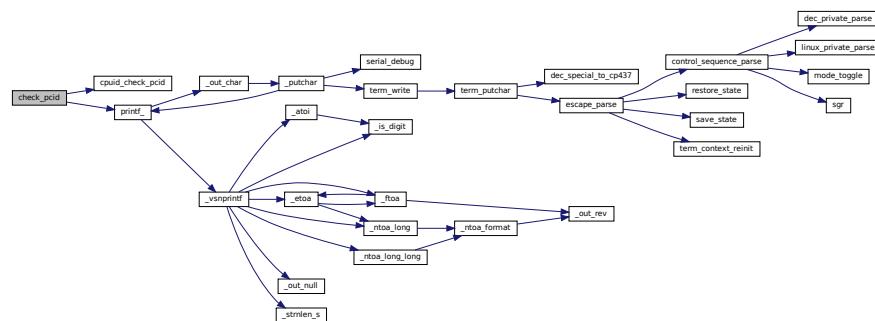
Check if CPUID is available.

**Returns**

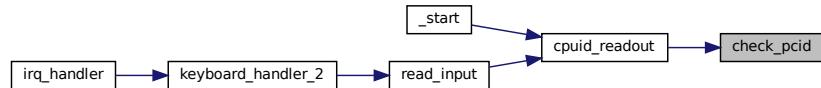
Yes if available No

Definition at line 288 of file [cpuUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.5.1.11 check\_sse()

```
void check_sse( )
```

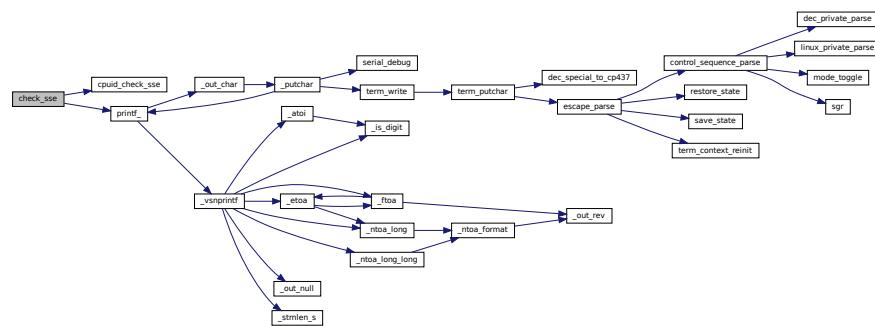
Check SSE extensions and print results if they are available.

##### Returns

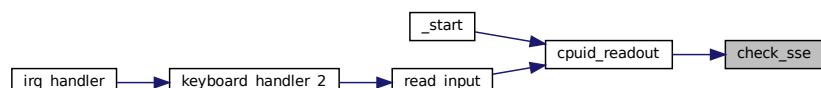
1 if SSE extensions are available 0

Definition at line 172 of file [cpuUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.5.1.12 check\_tm()

```
void check_tm ( )
```

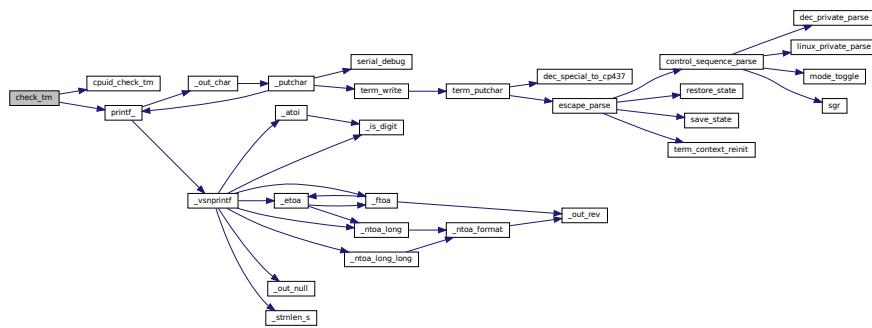
Check TM and print yes / no.

##### Returns

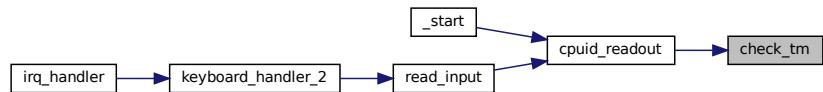
Yes if TM is set

Definition at line 433 of file [cpuUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



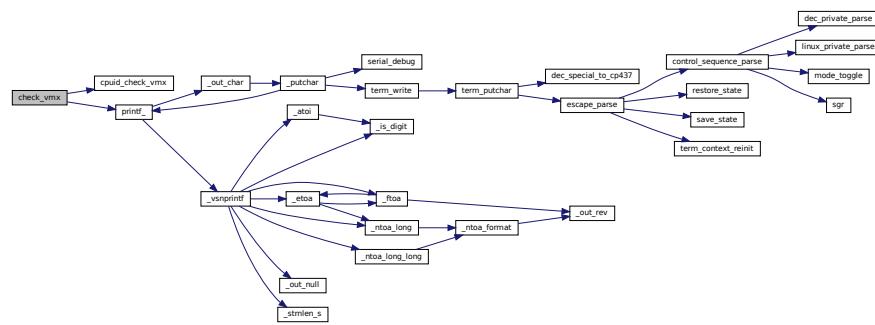
#### 4.5.1.13 check\_vmx()

```
void check_vmx ( )
```

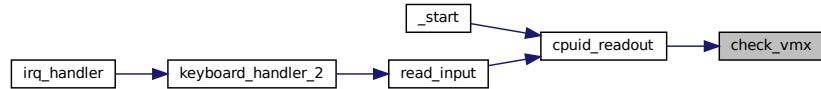
Check VMX is enabled or.

Definition at line 250 of file [cpuUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.5.1.14 `check_xsave()`

```
void check_xsave( )
```

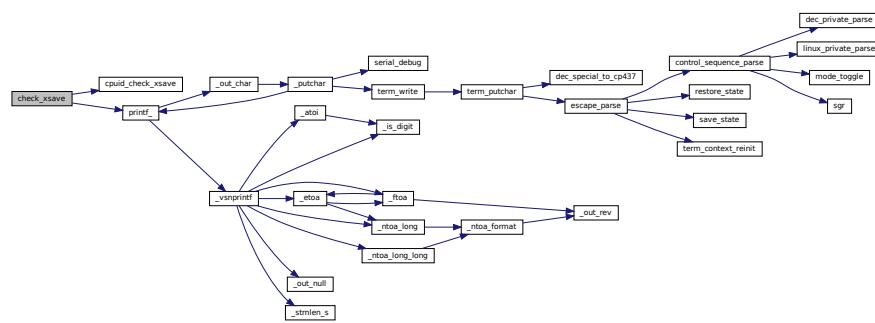
Check if XSAVE is available.

##### Returns

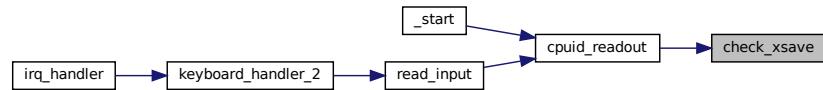
1 if XSAVE is

Definition at line 192 of file [cpuUtils.c](#).

Here is the call graph for this function:



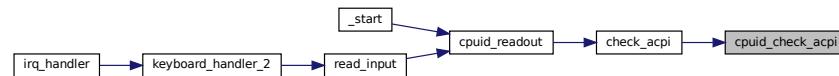
Here is the caller graph for this function:



#### 4.5.1.15 cpuid\_check\_acpi()

```
int cpuid_check_acpi ( )
```

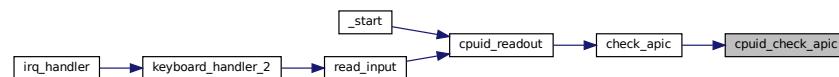
Here is the caller graph for this function:



#### 4.5.1.16 cpuid\_check\_apic()

```
int cpuid_check_apic ( )
```

Here is the caller graph for this function:



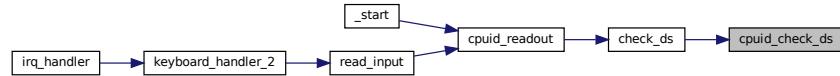
#### 4.5.1.17 cpuid\_check\_avx()

```
int cpuid_check_avx ( )
```

#### 4.5.1.18 cpuid\_check\_ds()

```
int cpuid_check_ds ( )
```

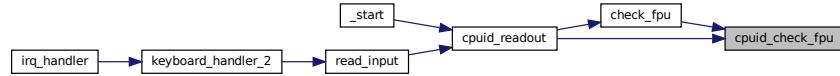
Here is the caller graph for this function:



#### 4.5.1.19 cpuid\_check\_fpu()

```
int cpuid_check_fpu ( )
```

Here is the caller graph for this function:



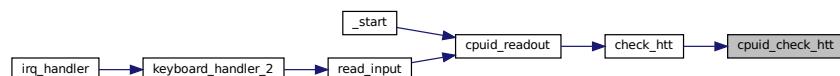
#### 4.5.1.20 cpuid\_check\_fxsr()

```
int cpuid_check_fxsr ( )
```

#### 4.5.1.21 cpuid\_check\_htt()

```
int cpuid_check_htt ( )
```

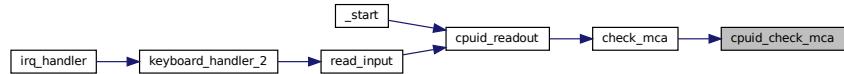
Here is the caller graph for this function:



#### 4.5.1.22 cpuid\_check\_mca()

```
int cpuid_check_mca ( )
```

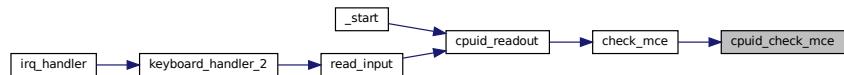
Here is the caller graph for this function:



#### 4.5.1.23 cpuid\_check\_mce()

```
int cpuid_check_mce ( )
```

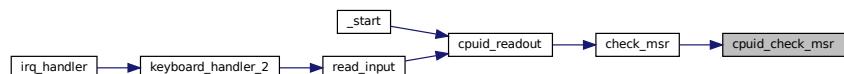
Here is the caller graph for this function:



#### 4.5.1.24 cpuid\_check\_msra()

```
int cpuid_check_msra ( )
```

Here is the caller graph for this function:



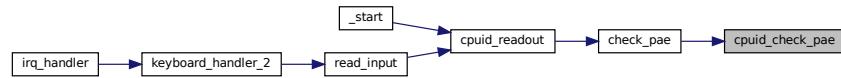
#### 4.5.1.25 cpuid\_check\_oxsave()

```
int cpuid_check_oxsave ( )
```

#### 4.5.1.26 cpuid\_check\_pae()

```
int cpuid_check_pae ( )
```

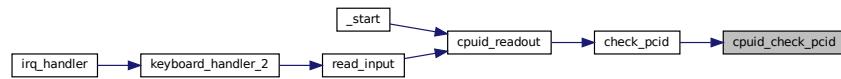
Here is the caller graph for this function:



#### 4.5.1.27 cpuid\_check\_pcid()

```
int cpuid_check_pcid ( )
```

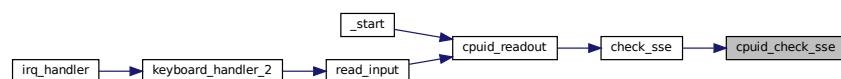
Here is the caller graph for this function:



#### 4.5.1.28 cpuid\_check\_sse()

```
int cpuid_check_sse ( )
```

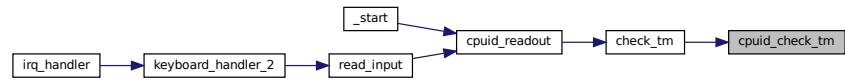
Here is the caller graph for this function:



#### 4.5.1.29 cpuid\_check\_tm()

```
int cpuid_check_tm ( )
```

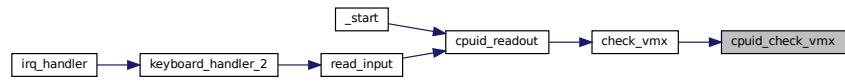
Here is the caller graph for this function:



#### 4.5.1.30 cpuid\_check\_vmx()

```
int cpuid_check_vmx ( )
```

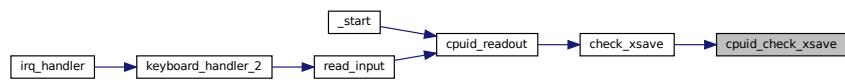
Here is the caller graph for this function:



#### 4.5.1.31 cpuid\_check\_xsave()

```
int cpuid_check_xsave ( )
```

Here is the caller graph for this function:



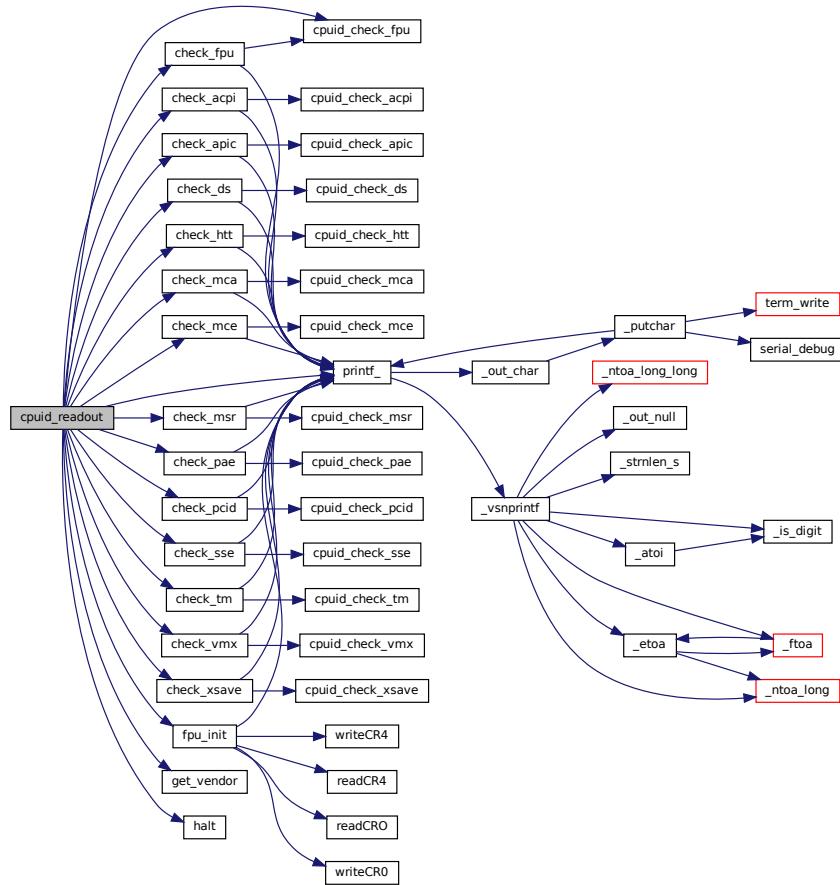
#### 4.5.1.32 cpuid\_readout()

```
void cpuid_readout ( )
```

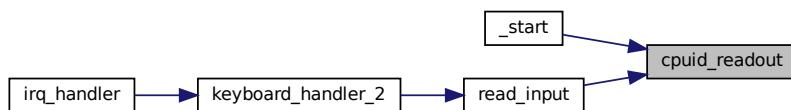
Print a message to the standard output. This is called from cpuid\_read ()

Definition at line 44 of file [cpuUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

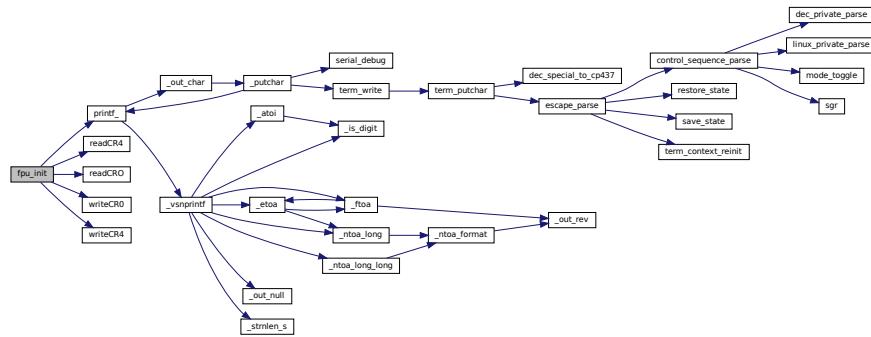


### 4.5.1.33 fpu\_init()

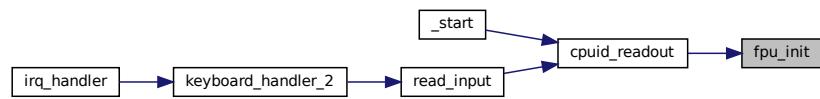
```
void fpu_init ( )
```

Definition at line 128 of file [cpuUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 4.5.1.34 get\_model()

```
int get_model (
    void )
```

Get the CPUID model.

#### Returns

The number of EBX

Definition at line 101 of file [cpuUtils.c](#).

#### 4.5.1.35 get\_vendor()

```
void get_vendor ( )
```

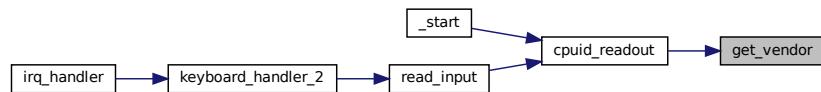
Get vendor information from CPUID.

##### Returns

0 on success - 1 on

Definition at line 112 of file [cpuUtils.c](#).

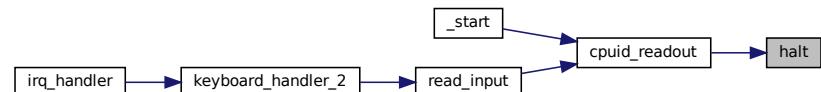
Here is the caller graph for this function:



#### 4.5.1.36 halt()

```
void halt ( )
```

Here is the caller graph for this function:



#### 4.5.1.37 no\_sse()

```
void no_sse ( )
```

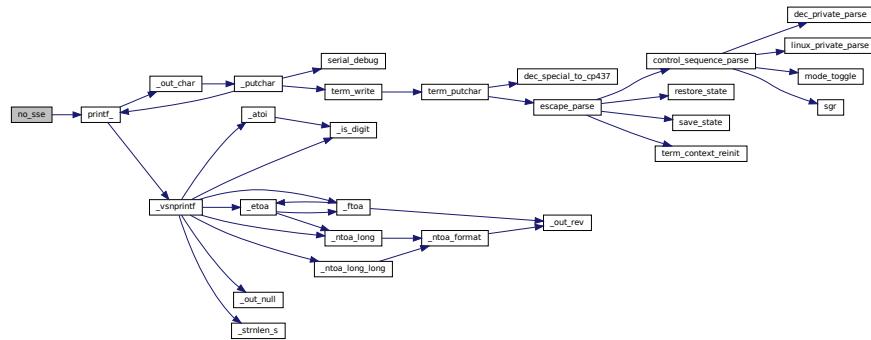
Print message to indicate SSE extensions are unavailable.

**Returns**

non - zero for success zero for

Definition at line 150 of file [cpuUtils.c](#).

Here is the call graph for this function:

**4.5.1.38 no\_xsave()**

```
void no_xsave( )
```

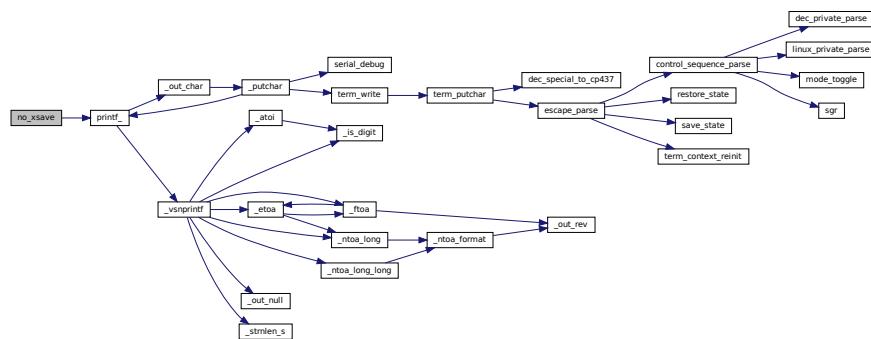
Display message to indicate XSAVE extensions are unavailable.

**Returns**

non - zero if message was displayed

Definition at line 161 of file [cpuUtils.c](#).

Here is the call graph for this function:



#### 4.5.1.39 vendor\_str1()

```
int vendor_str1 ( )
```

#### 4.5.1.40 vendor\_str2()

```
int vendor_str2 ( )
```

#### 4.5.1.41 vendor\_str3()

```
int vendor_str3 ( )
```

### 4.5.2 Variable Documentation

#### 4.5.2.1 CPU\_vendor

```
char CPU_vendor[13]
```

Definition at line [37](#) of file [cpuUtils.c](#).

#### 4.5.2.2 first\_run

```
bool first_run = true
```

Definition at line [39](#) of file [cpuUtils.c](#).

## 4.6 cpuUtils.c

```

00001 #include "include/cpuUtils.h"
00002 #include "include/printf.h"
00003 #include "include/registers.h"
00004 #include "include/global_defs.h"
00005 #include "include/limine.h"
00006 #include "include/memUtils.h"
00007 #include "include/paging/frameallocator.h"
00008 #include "include/paging/paging.h"
00009 #include "include/proc.h"
00010 #include "include/sched.h"
00011 #include "include/kernel.h"
00012 #include <cpuid.h>
00013 #include <stdbool.h>
00014
00015 extern int cpuid_check_sse();
00016 extern int cpuid_check_xsave();
00017 extern int cpuid_check_pcid();
00018 extern int cpuid_check_pae();
00019 extern int cpuid_check_mce();
00020 extern int cpuid_check_apic();
00021 extern int cpuid_check_mca();
00022 extern int cpuid_check_acpi();
00023 extern int cpuid_check_ds();
00024 extern int cpuid_check_tm();
00025 extern int cpuid_check_vmx();
00026 extern int cpuid_check_htt();
00027 extern int cpuid_check_fpu();
00028 extern int cpuid_check_msр();
00029 extern int cpuid_check_oxsave();
00030 extern int cpuid_check_avx();
00031 extern int cpuid_check_fxsr();
00032 extern int vendor_str1();
00033 extern int vendor_str2();
00034 extern int vendor_str3();
00035 extern void halt();
00036
00037 char CPU_vendor[13];
00038
00039 bool first_run = true;
00040
00041 void cpuid_readout()
00042 {
00043
00044     printf_( "%s\n", "-----");
00045     printf_( "%s\n", "|      CPUID Readout      |");
00046     printf_( "%s\n", "-----");
00047
00048     printf_( "%s\n", "-----");
00049     printf_( "%s\n", "|      Brand & Vendor    |");
00050     printf_( "%s\n", "-----");
00051
00052     get_vendor();
00053     printf_( "%s\n", "-----");
00054     printf_( "%s", "CPU Vendor: ");
00055     printf_( "%s\n", CPU_vendor);
00056     printf_( "%s\n", "-----");
00057     check_sse();
00058     check_xsave();
00059     check_xsave();
00060     check_pcid();
00061     check_pae();
00062     check_mce();
00063     check_apic();
00064     check_mca();
00065     check_acpi();
00066     check_ds();
00067     check_tm();
00068     check_vmx();
00069     check_htt();
00070     check_fpu();
00071     check_msр();
00072
00073     if (first_run == true)
00074     {
00075
00076         first_run = false;
00077
00078         if (cpuid_check_fpu() == 1)
00079         {
00080
00081             fpu_init();
00082         }
00083         else
00084         {
00085
00086             printf_( "%s\n", "ERROR: FUCK YOU NO FLOATS!");
00087             halt();
00088         }
00089

```

```
00089      }
00090      // halt();
00092
00093  printf_( "%s\n", "-----");
00094 }
00095
00096 /* Example: Get CPU's model number */
00101 int get_model(void)
00102 {
00103     int ebx, unused;
00104     __cpuid(0, unused, ebx, unused, unused);
00105     return ebx;
00106 }
00107
00112 void get_vendor()
00113 {
00114     uint32_t ebx, edx, ecx, eax;
00116
00117     __get_cpuid(0, &eax, &ebx, &ecx, &edx);
00118     for (size_t i = 0; i < 4; i++)
00119     {
00120         const size_t shiftor = i * 8;
00121         CPU_vendor[0 + i] = (ebx >> shiftor) & 0xFF;
00122         CPU_vendor[4 + i] = (edx >> shiftor) & 0xFF;
00123         CPU_vendor[8 + i] = (ecx >> shiftor) & 0xFF;
00124     }
00125     CPU_vendor[12] = 0;
00126 }
00127
00128 void fpu_init()
00129 {
00130     printf_( "%s\n", "INFO: Enabling the x87 FPU");
00132
00133     writeCR0(readCRO() | 1 << 1);
00134     writeCR0(readCRO() | 1 << 5);
00135     writeCR4(readCR4() | 1 << 9);
00136     writeCR4(readCR4() | 1 << 10);
00137     writeCR4(readCR4() | 1 << 18);
00138     cfg_XCR0();
00139     printf_( "%s", "XCR0: ");
00140     printf_( "0x%lx\n", read_XCR0());
00141     // writeCR0(readCRO() | 0 >> 2);
00142
00143     printf_( "%s\n", "INFO: x87 FPU now online");
00144 }
00145
00150 void no_sse()
00151 {
00152
00153     printf_( "%s\n", "SSE Extensions Unavailable.");
00154     printf_( "%s\n", "Floating Point Math will be offline.");
00155 }
00156
00161 void no_xsave()
00162 {
00163
00164     printf_( "%s\n", "XSAVE Extensions Unavailable.");
00165     printf_( "%s\n", "Floating Point Math will be offline.");
00166 }
00167
00172 void check_sse()
00173 {
00174
00175     int found = cpuid_check_sse();
00176
00177     if (found == 1)
00178     {
00179
00180         printf_( "%s\n", "SSE Extensions Available.");
00181
00182         printf_( "%s\n", "Enabling....");
00183
00184         printf_( "%s\n", "SSE Extensions Online!");
00185     }
00186 }
00187
00192 void check_xsave()
00193 {
00194
00195     int found = cpuid_check_xsave();
00196
00197     if (found == 1)
00198     {
00199
```

```
00200     printf_("\"%s\\n\", \"XSAVE Extensions Available.\");\n"
00201
00202     printf_("\"%s\\n\", \"Enabling....\");\n"
00203
00204     printf_("\"%s\\n\", \"XSAVE Extensions Online!\");\n"
00205
00206     printf_("\"%s\\n\", \"Floating Point Math Online using SSE and XSAVE!\");\n"
00207 }
00208 }
00209
00213 void check_fpu()
00214 {
00215
00216     if (cpuid_check_fpu() == 1)
00217     {
00218
00219         printf_("\"%s\\n\", \"x87 FPU: Yes\");\n"
00220     }
00221     else
00222     {
00223
00224         printf_("\"%s\\n\", \"x87 FPU: No\");\n"
00225     }
00226 }
00227
00232 void check_msr()
00233 {
00234
00235     if (cpuid_check_msr() == 1)
00236     {
00237
00238         printf_("\"%s\\n\", \"MSR: Yes\");\n"
00239     }
00240     else
00241     {
00242
00243         printf_("\"%s\\n\", \"MSR: No\");\n"
00244     }
00245 }
00246
00250 void check_vmx()
00251 {
00252
00253     if (cpuid_check_vmx() == 1)
00254     {
00255
00256         printf_("\"%s\\n\", \"VMX (HW Virtualization): Yes\");\n"
00257     }
00258     else
00259     {
00260
00261         printf_("\"%s\\n\", \"VMX (HW Virtualization): No\");\n"
00262     }
00263 }
00264
00269 void check_htt()
00270 {
00271
00272     if (cpuid_check_htt() == 1)
00273     {
00274
00275         printf_("\"%s\\n\", \"HTT: Yes\");\n"
00276     }
00277     else
00278     {
00279
00280         printf_("\"%s\\n\", \"HTT: No\");\n"
00281     }
00282 }
00283
00288 void check_pcid()
00289 {
00290
00291     if (cpuid_check_pcid() == 1)
00292     {
00293
00294         printf_("\"%s\\n\", \"PCID: Yes\");\n"
00295     }
00296     else
00297     {
00298
00299         printf_("\"%s\\n\", \"PCID: No\");\n"
00300     }
00301 }
00302
00307 void check_pae()
00308 {
```

```
00309     int found = cpuid_check_pae();
00310
00311     if (found == 1)
00312     {
00313
00314         printf_( "%s\n", "PAE: Yes");
00315     }
00316     else
00317     {
00318
00319         printf_( "%s\n", "PAE: No");
00320     }
00321 }
00322 }
00323
00328 void check_mce()
00329 {
00330
00331     int found = cpuid_check_mce();
00332
00333     if (found == 1)
00334     {
00335
00336         printf_( "%s\n", "MCE: Yes");
00337     }
00338     else
00339     {
00340
00341         printf_( "%s\n", "MCE: No");
00342     }
00343 }
00344
00349 void check_apic()
00350 {
00351
00352     int found = cpuid_check_apic();
00353
00354     if (found == 1)
00355     {
00356
00357         printf_( "%s\n", "APIC: Yes");
00358     }
00359     else
00360     {
00361
00362         printf_( "%s\n", "APIC: No");
00363     }
00364 }
00365
00370 void check_mca()
00371 {
00372
00373     int found = cpuid_check_mca();
00374
00375     if (found == 1)
00376     {
00377
00378         printf( "%s\n", "MCA: Yes");
00379     }
00380     else
00381     {
00382
00383         printf_( "%s\n", "MCA: No");
00384     }
00385 }
00386
00391 void check_acpi()
00392 {
00393
00394     int found = cpuid_check_acpi();
00395
00396     if (found == 1)
00397     {
00398
00399         printf_( "%s\n", "ACPI: Yes");
00400     }
00401     else
00402     {
00403
00404         printf_( "%s\n", "ACPI: No");
00405     }
00406 }
00407
00412 void check_ds()
00413 {
00414
00415     int found = cpuid_check_ds();
```

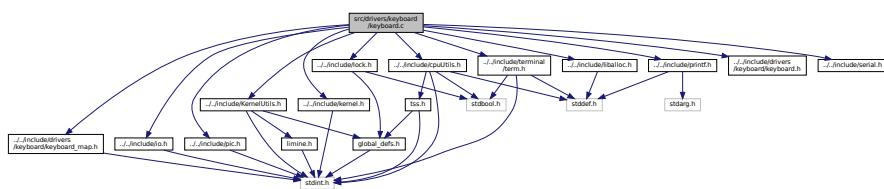
```

00416
00417     if (found == 1)
00418     {
00419
00420         printf_(""%s\n", "DS: Yes");
00421     }
00422     else
00423     {
00424
00425         printf_(""%s\n", "DS: No");
00426     }
00427 }
00428
00429 void check_tm()
00430 {
00431
00432     int found = cpuid_check_tm();
00433
00434     if (found == 1)
00435     {
00436
00437         printf_(""%s\n", "TM: Yes");
00438     }
00439     else
00440     {
00441
00442         printf_(""%s\n", "TM: No");
00443     }
00444 }
00445
00446 }
00447 }
00448 }
```

## 4.7 src/drivers/keyboard/keyboard.c File Reference

```
#include "../../include/drivers/keyboard/keyboard_map.h"
#include "../../include/drivers/keyboard/keyboard.h"
#include "../../include/io.h"
#include "../../include/pic.h"
#include "../../include/printf.h"
#include "../../include/KernelUtils.h"
#include "../../include/terminal/term.h"
#include "../../include/liballoc.h"
#include "../../include/serial.h"
#include "../../include/lock.h"
#include "../../include/kernel.h"
#include "../../include/cpuUtils.h"
```

Include dependency graph for keyboard.c:



## Macros

- `#define KBD_STACK_SIZE 255`

## Functions

- `void kbd_push (char data)`

- char [kbd\\_pop\(\)](#)  
*Push a character onto the keyboard stack.*
- void [keyboard\\_handler\(\)](#)  
*Pop a character from the kbd stack.*
- void [keyboard\\_handler\\_2\(\)](#)  
*This is the keyboard interrupt handler. It reads the status and pushes the keycode to the keyboard stack.*
- void [keyboard\\_init\(\)](#)  
*Keyboard handler for 2.0 and later.*
- void [read\\_input\(\)](#)  
*Read input from the keyboard.*
- char [k\\_getchar\(\)](#)  
*Get a character from k\_char.*
- void [keyboard\\_init\(\)](#)  
*Initialize the keyboard. Unmask IRQ and read keyboard data.*

## Variables

- char [k\\_char](#)
- char [kbd\\_stack\[KBD\\_STACK\\_SIZE\]](#)
- int [kbd\\_top = -1](#)
- static [spinlock\\_t reader\\_lock = SPINLOCK\\_INIT](#)

### 4.7.1 Macro Definition Documentation

#### 4.7.1.1 KBD\_STACK\_SIZE

```
#define KBD_STACK_SIZE 255
```

Definition at line 14 of file [keyboard.c](#).

### 4.7.2 Function Documentation

#### 4.7.2.1 k\_getchar()

```
char k_getchar( )
```

Get a character from k\_char.

##### Returns

The character or 0 if none

Definition at line 224 of file [keyboard.c](#).

### 4.7.2.2 kbd\_pop()

```
char kbd_pop ( )
```

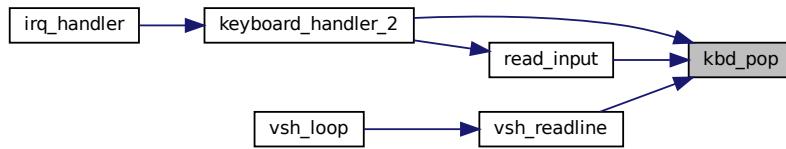
Pop a character from the kbd stack.

#### Returns

character that was popped

Definition at line 52 of file [keyboard.c](#).

Here is the caller graph for this function:



### 4.7.2.3 kbd\_push()

```
void kbd_push (
    char data )
```

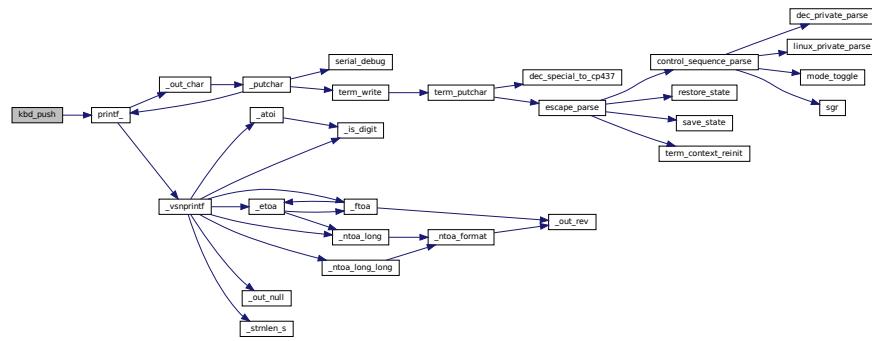
Push a character onto the keyboard stack.

#### Parameters

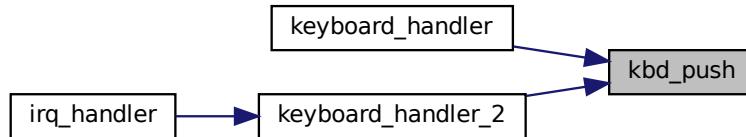
<i>data</i>	The character to push onto
-------------	----------------------------

Definition at line 28 of file [keyboard.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.7.2.4 keyboard\_handler()

```
void keyboard_handler ( )
```

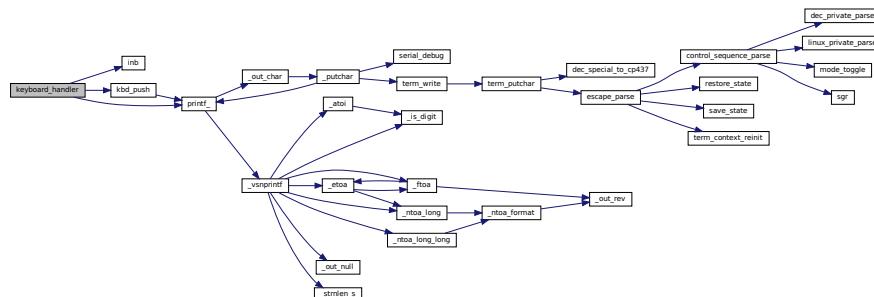
This is the keyboard interrupt handler. It reads the status and pushes the keycode to the keyboard stack.

##### Returns

Returns nothing. If there is an error it returns

Definition at line 77 of file [keyboard.c](#).

Here is the call graph for this function:



#### 4.7.2.5 keyboard\_handler\_2()

```
void keyboard_handler_2 ( )
```

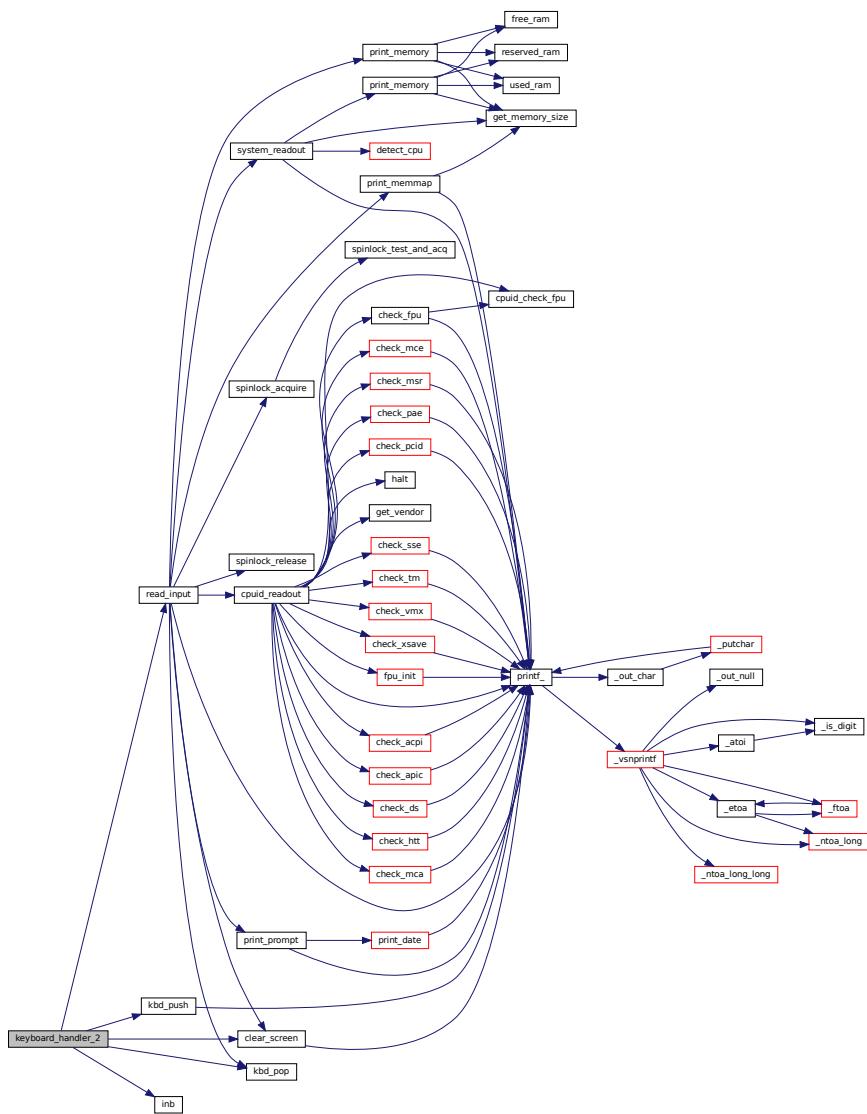
Keyboard handler for 2. 0 and later.

##### Returns

none

Definition at line 135 of file [keyboard.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.7.2.6 keyboard\_init()

```
void keyboard_init ( )
```

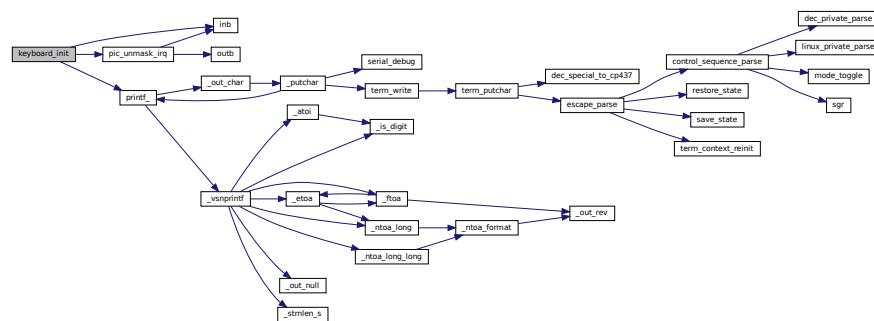
Initialize the keyboard. Unmask IRQ and read keyboard data.

##### Returns

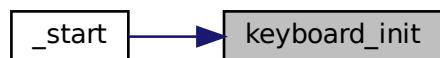
0 on success non - zero

Definition at line 243 of file [Keyboard.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



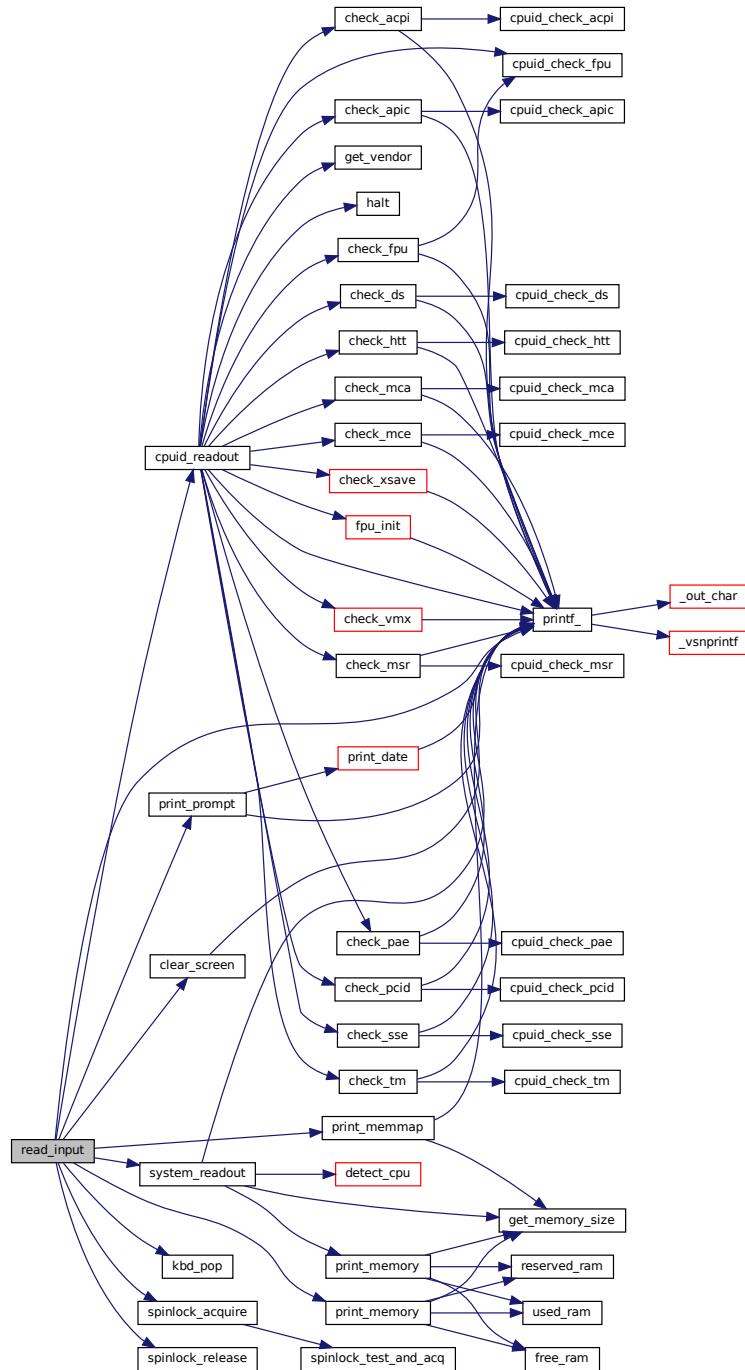
#### 4.7.2.7 read\_input()

```
void read_input ( )
```

Read input from the keyboard.

Definition at line 173 of file [keyboard.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 4.7.3 Variable Documentation

#### 4.7.3.1 k\_char

```
char k_char
```

Definition at line 16 of file [keyboard.c](#).

#### 4.7.3.2 kbd\_stack

```
char kbd_stack[KBD_STACK_SIZE]
```

Definition at line 18 of file [keyboard.c](#).

#### 4.7.3.3 kbd\_top

```
int kbd_top = -1
```

Definition at line 20 of file [keyboard.c](#).

#### 4.7.3.4 reader\_lock

```
spinlock_t reader_lock = SPINLOCK_INIT [static]
```

Definition at line 22 of file [keyboard.c](#).

## 4.8 keyboard.c

```
00001 #include "../../../include/drivers/keyboard/keyboard_map.h"
00002 #include "../../../include/drivers/keyboard/keyboard.h"
00003 #include "../../../include/io.h"
00004 #include "../../../include/pic.h"
00005 #include "../../../include/printf.h"
00006 #include "../../../include/KernelUtils.h"
00007 #include "../../../include/terminal/term.h"
00008 #include "../../../include/liballoc.h"
00009 #include "../../../include/serial.h"
00010 #include "../../../include/lock.h"
00011 #include "../../../include/kernel.h"
00012 #include "../../../include/cpuUtils.h"
00013
00014 #define KBD_STACK_SIZE 255
00015
00016 char k_char;
00017
00018 char kbd_stack[KBD_STACK_SIZE];
00019
00020 int kbd_top = -1;
00021
00022 static spinlock_t reader_lock = SPINLOCK_INIT;
00023
00028 void kbd_push(char data)
00029 {
00030
00031     if (kbd_top == KBD_STACK_SIZE - 1)
00032     {
00033
00034         kerror_mode = 1;
00035
00036         printf_(""%s\n", "ERROR: Keyboard stack overflow!");
00037
00038         kerror_mode = 0;
00039     }
00040     else
00041     {
00042
00043         kbd_top++;
00044         kbd_stack[kbd_top] = data;
00045     }
00046 }
00047
00052 char kbd_pop()
00053 {
00054
00055     int data;
00056
00057     if (kbd_top == -1)
00058     {
00059
00060         return 0;
00061     }
00062     else
00063     {
00064
00065         data = kbd_stack[kbd_top];
00066
00067         kbd_top--;
00068
00069         return data;
00070     }
00071 }
00072
00077 void keyboard_handler()
00078 {
00079
00080     uint8_t status = inb(KEYBOARD_STATUS_PORT);
00081
00082 //    printf_("0x%llx\n", status);
00083
00084     if (status & 0x1)
00085     {
00086
00087         char keycode = inb(KEYBOARD_DATA_PORT);
00088
00089         inb(KEYBOARD_DATA_PORT);
00090
00091         kbd_push(keyboard_map[keycode]);
00092
00093         if (keycode < 0 || keycode >= 128)
00094         {
00095             return;
00096         }
00097 }
```

```

00098     // printf_("0x%llx\n", keyboard_map[keycode]);
00099     // printf_("0x%llx\n", keycode);
00100
00101     if (keycode == 0x1c)
00102     {
00103         k_char = '+';
00104     }
00105     else
00106     {
00107
00108         k_char = keyboard_map[keycode];
00109     }
00110
00111     // printf_("%s\n", &k_char);
00112
00113     if (term_context)
00114     {
00115
00116         // term_write(term_context, &keyboard_map[keycode], sizeof(char));
00117     }
00118     else
00119     {
00120
00121         printf_("ERROR: TERMINAL WRITE FAILURE!");
00122
00123         return;
00124     }
00125
00126     return;
00127 }
00128 }
00129
00130 void keyboard_handler_2()
00131 {
00132
00133     uint8_t status = inb(KEYBOARD_STATUS_PORT);
00134
00135     if (status & 0x1)
00136     {
00137         char keycode = inb(KEYBOARD_DATA_PORT);
00138         inb(KEYBOARD_DATA_PORT);
00139
00140         if (keycode < 0 || keycode >= 128)
00141         {
00142             return;
00143         }
00144
00145         kbd_push(keyboard_map[keycode]);
00146
00147         if (keycode == 28)
00148         {
00149
00150             kbd_pop();
00151
00152             kbd_push(28);
00153
00154         }
00155
00156         clear_screen();
00157
00158         // printf_("%i\n", keycode);
00159
00160         read_input();
00161
00162         // printf_("%s\n", "Test");
00163
00164     }
00165 }
00166
00167 }
00168
00169
00170 void read_input()
00171 {
00172
00173     spinlock_acquire(&reader_lock);
00174
00175     char option = kbd_pop();
00176
00177     if (option == 28)
00178     {
00179         clear_screen();
00180         print_prompt();
00181     }
00182
00183     switch (option)
00184     {
00185
00186         case '1':
00187             print_memmap();
00188             printf_("Press Enter to Return to The Menu.");
00189
00190
00191

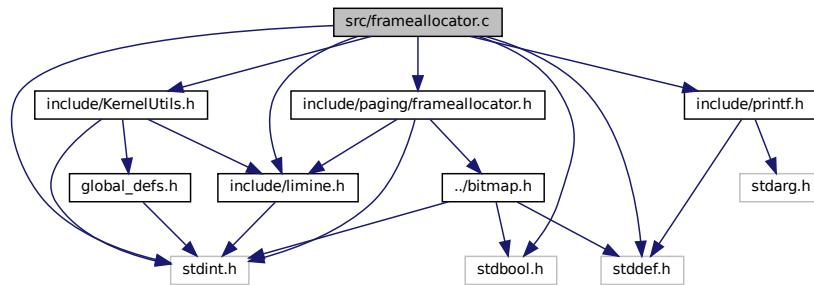
```

```
00192         break;
00193
00194     case '2':
00195         printf_("\"%s\\n\", \"I suggest you check the results with the Intel and AMD dev manuals.\");
00196         cpuid_readout();
00197         printf_("\"%s\\n\", \"Press Enter to Return to The Menu.\");
00198         break;
00199
00200     case '3':
00201         print_memory();
00202         printf_("\"%s\\n\", \"Press Enter to Return to The Menu.\");
00203         break;
00204
00205     case '4':
00206         system_readout();
00207         printf_("\"%s\\n\", \"Press Enter to Return to The Menu.\");
00208         break;
00209
00210     default:
00211         clear_screen();
00212         print_prompt();
00213     }
00214
00215     spinlock_release(&reader_lock);
00216
00217 // printf_("\"%s\\n\", &option);
00218 }
00219
00220 char k_getchar()
00221 {
00222     char c;
00223
00224     while (k_char <= 0)
00225     {
00226
00227         c = k_char;
00228         k_char = 0;
00229     }
00230
00231     return c;
00232 }
00233
00234 void keyboard_init()
00235 {
00236
00237     pic_unmask_irq(1);
00238     inb(KEYBOARD_DATA_PORT);
00239     inb(KEYBOARD_DATA_PORT);
00240
00241     printf_("\"%s\\n\", \"Keyboard Init\");
00242 }
```

## 4.9 src/frameallocator.c File Reference

```
#include "include/limine.h"
#include <stdint.h>
#include "include/paging/frameallocator.h"
#include <stdbool.h>
#include <stddef.h>
#include "include/KernelUtils.h"
#include "include/printf.h"
```

Include dependency graph for frameallocator.c:



## Functions

- `void halt ()`
- `const char * MemoryMapTypeString (int type)`
- `void read_memory_map ()`
- `void * frame_request ()`
- `void print_frame_bitmap ()`
- `void * frame_request_multiple (uint32_t count)`
- `void frame_free (void *address)`
- `void frame_free_multiple (void *address, uint64_t pageCount)`
- `void frame_lock (void *address)`
- `void frame_lock_multiple (void *address, uint64_t pageCount)`
- `uint64_t free_ram ()`
- `uint64_t used_ram ()`
- `uint64_t reserved_ram ()`

## Variables

- `uint64_t freeMemory = 0`
- `uint64_t reservedMemory = 0`
- `uint64_t usedMemory = 0`
- `bool initialized = false`
- `uint8_t * frameBitmap = NULL`
- `uint64_t bitmapSize = 0`

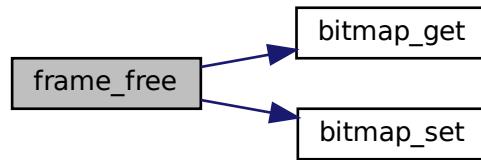
### 4.9.1 Function Documentation

#### 4.9.1.1 frame\_free()

```
void frame_free (
    void * address )
```

Definition at line 207 of file [frameallocator.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

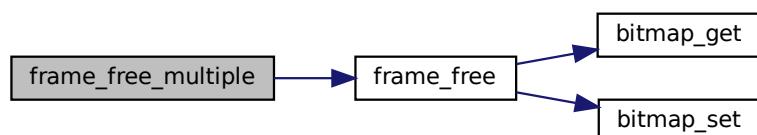


#### 4.9.1.2 frame\_free\_multiple()

```
void frame_free_multiple (
    void * address,
    uint64_t pageCount )
```

Definition at line 222 of file [frameallocator.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

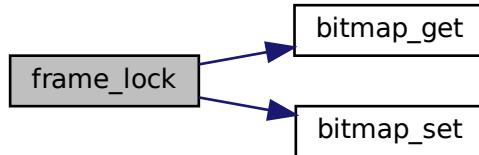


#### 4.9.1.3 frame\_lock()

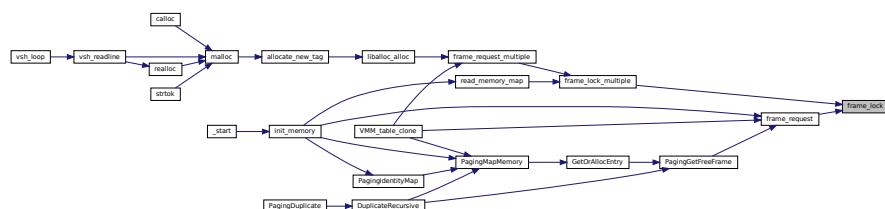
```
void frame_lock (
    void * address )
```

Definition at line 230 of file [frameallocator.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

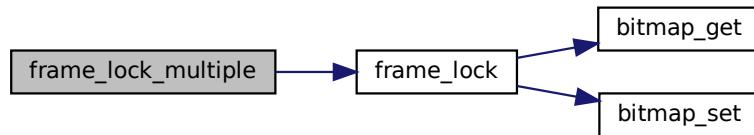


#### 4.9.1.4 frame\_lock\_multiple()

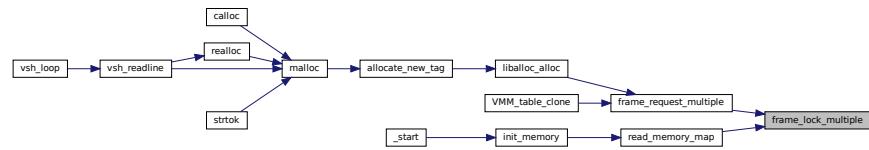
```
void frame_lock_multiple (
    void * address,
    uint64_t pageCount )
```

Definition at line 245 of file [frameallocator.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

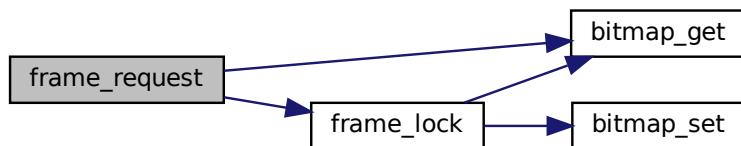


#### 4.9.1.5 frame\_request()

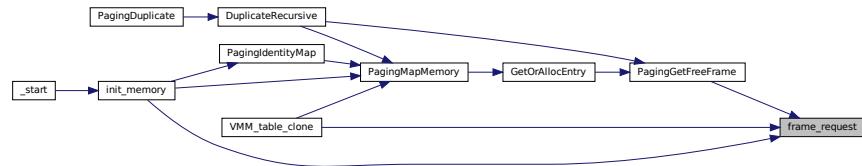
```
void* frame_request ( )
```

Definition at line 146 of file [frameallocator.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

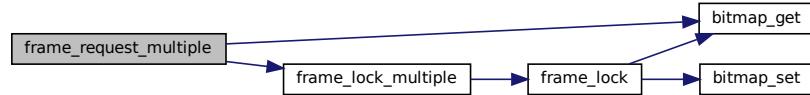


#### 4.9.1.6 frame\_request\_multiple()

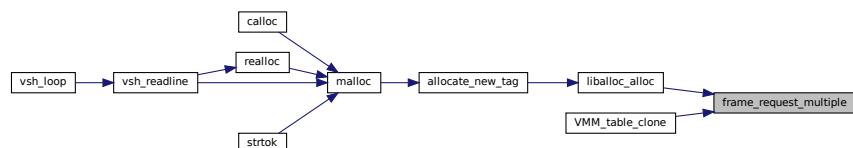
```
void* frame_request_multiple (
    uint32_t count )
```

Definition at line 179 of file [frameallocator.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

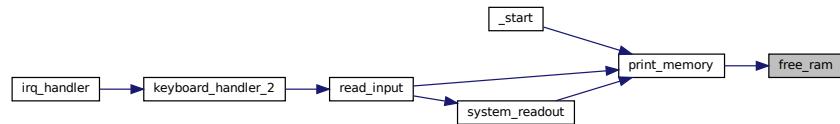


#### 4.9.1.7 free\_ram()

```
uint64_t free_ram ( )
```

Definition at line 253 of file [frameallocator.c](#).

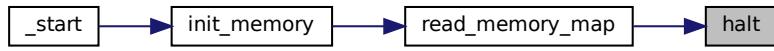
Here is the caller graph for this function:



#### 4.9.1.8 halt()

```
void halt ( )
```

Here is the caller graph for this function:



#### 4.9.1.9 MemoryMapTypeString()

```
const char* MemoryMapTypeString (
    int type )
```

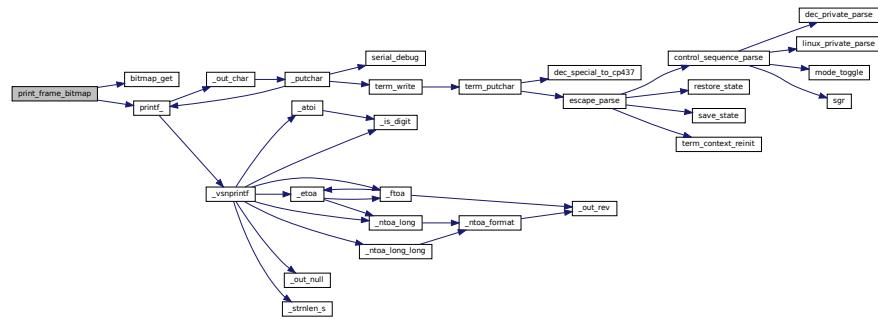
Definition at line 18 of file [frameallocator.c](#).

#### 4.9.1.10 print\_frame\_bitmap()

```
void print_frame_bitmap ( )
```

Definition at line 161 of file [frameallocator.c](#).

Here is the call graph for this function:

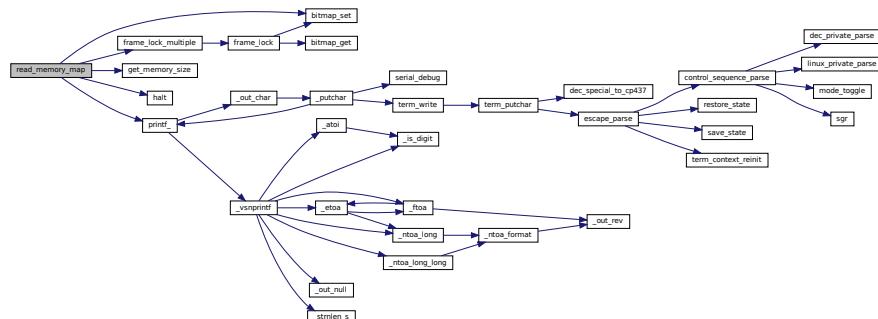


#### 4.9.1.11 `read_memory_map()`

```
void read_memory_map ( )
```

Definition at line 60 of file [frameallocator.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

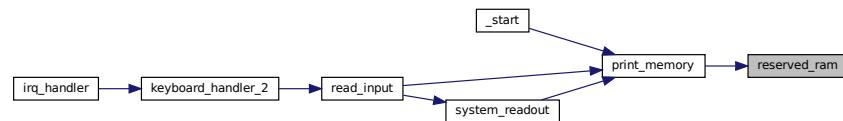


#### 4.9.1.12 reserved\_ram()

```
uint64_t reserved_ram ( )
```

Definition at line 263 of file [frameallocator.c](#).

Here is the caller graph for this function:

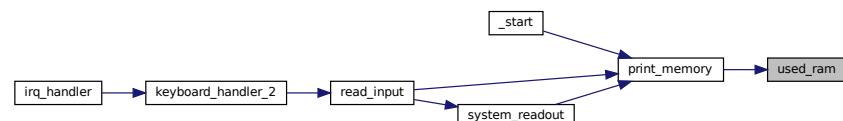


#### 4.9.1.13 used\_ram()

```
uint64_t used_ram ( )
```

Definition at line 258 of file [frameallocator.c](#).

Here is the caller graph for this function:



## 4.9.2 Variable Documentation

### 4.9.2.1 bitmapSize

```
uint64_t bitmapSize = 0
```

Definition at line 14 of file [frameallocator.c](#).

#### 4.9.2.2 frameBitmap

```
uint8_t* frameBitmap = NULL
```

Definition at line 13 of file [frameallocator.c](#).

#### 4.9.2.3 freeMemory

```
uint64_t freeMemory = 0
```

Definition at line 9 of file [frameallocator.c](#).

#### 4.9.2.4 initialized

```
bool initialized = false
```

Definition at line 12 of file [frameallocator.c](#).

#### 4.9.2.5 reservedMemory

```
uint64_t reservedMemory = 0
```

Definition at line 10 of file [frameallocator.c](#).

#### 4.9.2.6 usedMemory

```
uint64_t usedMemory = 0
```

Definition at line 11 of file [frameallocator.c](#).

## 4.10 frameallocator.c

```

00001 #include "include/limine.h"
00002 #include <stdint.h>
00003 #include "include/paging/frameallocator.h"
00004 #include <stdbool.h>
00005 #include <stddef.h>
00006 #include "include/KernelUtils.h"
00007 #include "include/printf.h"
00008
00009 uint64_t freeMemory = 0;
00010 uint64_t reservedMemory = 0;
00011 uint64_t usedMemory = 0;
00012 bool initialized = false;
00013 uint8_t *frameBitmap = NULL;
00014 uint64_t bitmapSize = 0;
00015
00016 extern void halt();
00017
00018 const char *MemoryMapTypeString(int type)
00019 {
00020     switch (type)
00021     {
00022         case LIMINE_MEMMAP_USABLE:
00023             return "Usable";
00025
00026         case LIMINE_MEMMAP_RESERVED:
00027             return "Reserved";
00029
00030         case LIMINE_MEMMAP_ACPI_RECLAIMABLE:
00031             return "ACPI Reclaimable";
00033
00034         case LIMINE_MEMMAP_ACPI_NVS:
00035             return "ACPI NVS";
00037
00038         case LIMINE_MEMMAP_BAD_MEMORY:
00039             return "Bad Memory";
00041
00042         case LIMINE_MEMMAP_BOOTLOADER_RECLAIMABLE:
00043             return "Bootloader Reclaimable";
00045
00046         case LIMINE_MEMMAP_KERNEL_AND_MODULES:
00047             return "Kernel and Modules";
00049
00050         case LIMINE_MEMMAP_FRAMEBUFFER:
00051             return "Framebuffer";
00053
00054     default:
00055         return "Unknown";
00057     }
00058 }
00059
00060 void read_memory_map()
00061 {
00062     if (initialized)
00063         return;
00064
00065     initialized = true;
00066
00067     void *largestFreeMemSegment = NULL;
00068     size_t largestFreeMemSegmentSize = 0;
00069     uint64_t bitmap_entry;
00070
00071     uint64_t memorySize = get_memory_size(); // I think this will work the same?
00072     freeMemory = memorySize;
00073
00074     bitmapSize = memorySize / 0x1000 / 8 + 1;
00075
00076     printf_("%s", "Bitmap Size is: ");
00077     printf_("0x%llx\n", bitmapSize);
00078
00079     for (uint64_t i = 0; i < memmap_req.response->entry_count; i++)
00080     {
00081         struct limine_memmap_entry *entry = memmap_req.response->entries[i];
00082
00083         if (entry->type == LIMINE_MEMMAP_USABLE && entry->length > bitmapSize)
00084         {
00085             largestFreeMemSegment = (void *)entry->base;

```

```

00086         largestFreeMemSegmentSize = entry->length;
00087         bitmap_entry = i;
00088     }
00089 }
00090
00091 if (largestFreeMemSegment == NULL)
00092 {
00093     printf_( "%s\n", "!!!Kernel Panic!!!");
00094     printf_( "%s\n", "No Suitable Memory Map Entries Found!" );
00095     printf_( "%s\n", "!!!Kernel Panic!!!");
00096     halt();
00097 }
00098
00099 printf_( "%s", "Address of Largest Free Entry: " );
00100 printf_( "0x%llx\n", (uint64_t)largestFreeMemSegment );
00101
00102 printf_( "%s", "Size of Largest Free Entry: " );
00103 printf_( "0x%llx\n", largestFreeMemSegmentSize );
00104
00105 memmap_req.response->entries[bitmap_entry]->length -= bitmapSize;
00106
00107 largestFreeMemSegmentSize = memmap_req.response->entries[bitmap_entry]->length;
00108
00109 printf_( "%s", "Size of Largest Free Entry Post Bitmap Allocation: " );
00110 printf_( "0x%llx\n", largestFreeMemSegmentSize );
00111
00112 // print_memmap();
00113
00114 frameBitmap = (uint8_t *)largestFreeMemSegment;
00115
00116 for (uint64_t i = 0; i < bitmapSize * 8; i++)
00117 {
00118     bitmap_set(frameBitmap, i, true);
00119
00120     freeMemory -= 0x1000;
00121     reservedMemory += 0x1000;
00122 }
00123
00124 for (uint64_t i = 0; i < memmap_req.response->entry_count; i++)
00125 {
00126     struct limine_memmap_entry *entry = memmap_req.response->entries[i];
00127
00128     if (entry->type == LIMINE_MEMMAP_USABLE)
00129     {
00130         printf("Unlocking usable pages for %p-%p (%llu pages)\n", entry->base, entry->base +
entry->length, entry->length / 0x1000);
00131
00132         for (uint64_t index = 0, startIndex = entry->base / 0x1000;
00133             index < entry->length / 0x1000; index++)
00134         {
00135             bitmap_set(frameBitmap, index + startIndex, false);
00136
00137             freeMemory += 0x1000;
00138             reservedMemory -= 0x1000;
00139         }
00140     }
00141 }
00142
00143 frame_lock_multiple(frameBitmap, bitmapSize / 0x1000 + 1);
00144 }
00145
00146 void *frame_request()
00147 {
00148     for (uint64_t i = 0; i < bitmapSize * 8; i++)
00149     {
00150         if (bitmap_get(frameBitmap, i) == true)
00151             continue;
00152
00153         frame_lock((void*)(i * 0x1000));
00154
00155         return (void*)(i * 0x1000);
00156     }
00157
00158     return NULL; // Page Frame Swap to file
00159 }
00160
00161 void print_frame_bitmap()
00162 {
00163
00164     printf_( "%s\n", "Frame Bitmap Readout Start" );
00165     printf_( "%s\n", "-----" );
00166
00167     for (uint64_t i = 0; i < bitmapSize * 8; i++)
00168     {
00169         printf_( "%s", "Data For Frame Bitmap Entry: " );
00170         printf_( "0x%llx\n", i );
00171

```

```
00172     printf_("\"%s\", \"Data: \"");
00173     printf_("0x%llx\n", bitmap_get(frameBitmap, i));
00174 }
00175
00176 printf_("%s\n", "-----");
00177 }
00178
00179 void *frame_request_multiple(uint32_t count)
00180 {
00181     uint32_t freeCount = 0;
00182
00183     for (uint32_t i = 0; i < bitmapSize * 8; i++)
00184     {
00185         if (bitmap_get(frameBitmap, i) == false)
00186         {
00187             if (freeCount == count)
00188             {
00189                 uint32_t index = i - count;
00190
00191                 frame_lock_multiple((void *)((uint64_t)index * 0x1000), count);
00192
00193                 return (void *)((uint64_t)index * 0x1000);
00194             }
00195
00196             freeCount++;
00197         }
00198         else
00199         {
00200             freeCount = 0;
00201         }
00202     }
00203
00204     return NULL;
00205 }
00206
00207 void frame_free(void *address)
00208 {
00209     uint64_t index = (uint64_t)address / 0x1000;
00210
00211     if (bitmap_get(frameBitmap, index) == false)
00212     {
00213         return;
00214     }
00215
00216     bitmap_set(frameBitmap, index, false);
00217
00218     freeMemory += 0x1000;
00219     usedMemory -= 0x1000;
00220 }
00221
00222 void frame_free_multiple(void *address, uint64_t pageCount)
00223 {
00224     for (uint64_t t = 0; t < pageCount; t++)
00225     {
00226         frame_free((void *)((uint64_t)address + (t * 0x1000)));
00227     }
00228 }
00229
00230 void frame_lock(void *address)
00231 {
00232     uint64_t index = (uint64_t)address / 0x1000;
00233
00234     if (bitmap_get(frameBitmap, index) == true)
00235     {
00236         return;
00237     }
00238
00239     bitmap_set(frameBitmap, index, true);
00240
00241     freeMemory -= 0x1000;
00242     usedMemory += 0x1000;
00243 }
00244
00245 void frame_lock_multiple(void *address, uint64_t pageCount)
00246 {
00247     for (uint64_t t = 0; t < pageCount; t++)
00248     {
00249         frame_lock((void *)((uint64_t)address + (t * 0x1000)));
00250     }
00251 }
00252
00253 uint64_t free_ram()
00254 {
00255     return freeMemory;
00256 }
00257
00258 uint64_t used_ram()
```

```

00259 {
00260     return usedMemory;
00261 }
00262
00263 uint64_t reserved_ram()
00264 {
00265     return reservedMemory;
00266 }

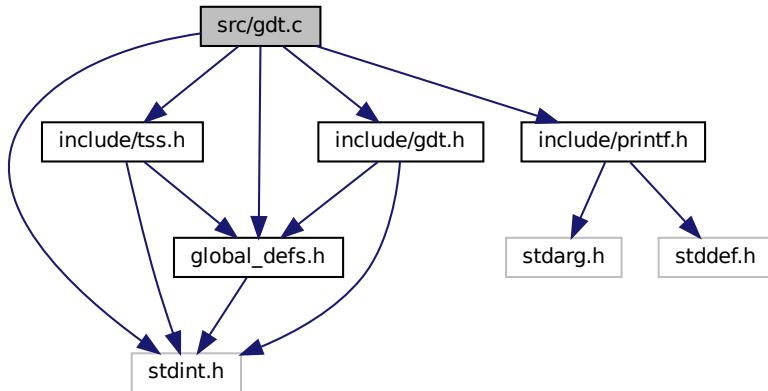
```

## 4.11 src/gdt.c File Reference

```

#include "include/gdt.h"
#include "include/global_defs.h"
#include <stdint.h>
#include "include/printf.h"
#include "include/tss.h"
Include dependency graph for gdt.c:

```



## Functions

- void `breakpoint()`
- void `serial_debug(int)`
- void `halt()`
- void `LoadGDT_Stage1()`

*Stage 1 of GDT loading.*

## Variables

- `uint8_t TssStack [0x100000]`
- `uint8_t ist1Stack [0x100000]`
- `uint8_t ist2Stack [0x100000]`
- `uint64_t rsp0`
- `struct TSS tss = {0}`
- `ALIGN_4K struct GDT gdt`
- `struct GDT_Desc desc`

## 4.11.1 Function Documentation

### 4.11.1.1 breakpoint()

```
void breakpoint ( )
```

### 4.11.1.2 halt()

```
void halt ( )
```

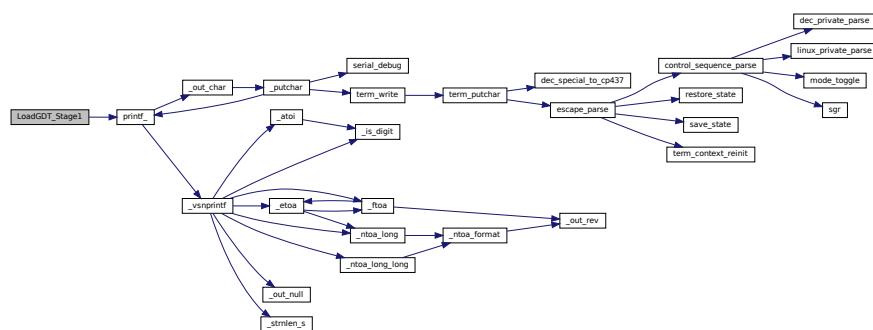
### 4.11.1.3 LoadGDT\_Stage1()

```
void LoadGDT_Stage1 ( )
```

Stage 1 of [GDT](#) loading.

Definition at line 118 of file [gdt.c](#).

Here is the call graph for this function:



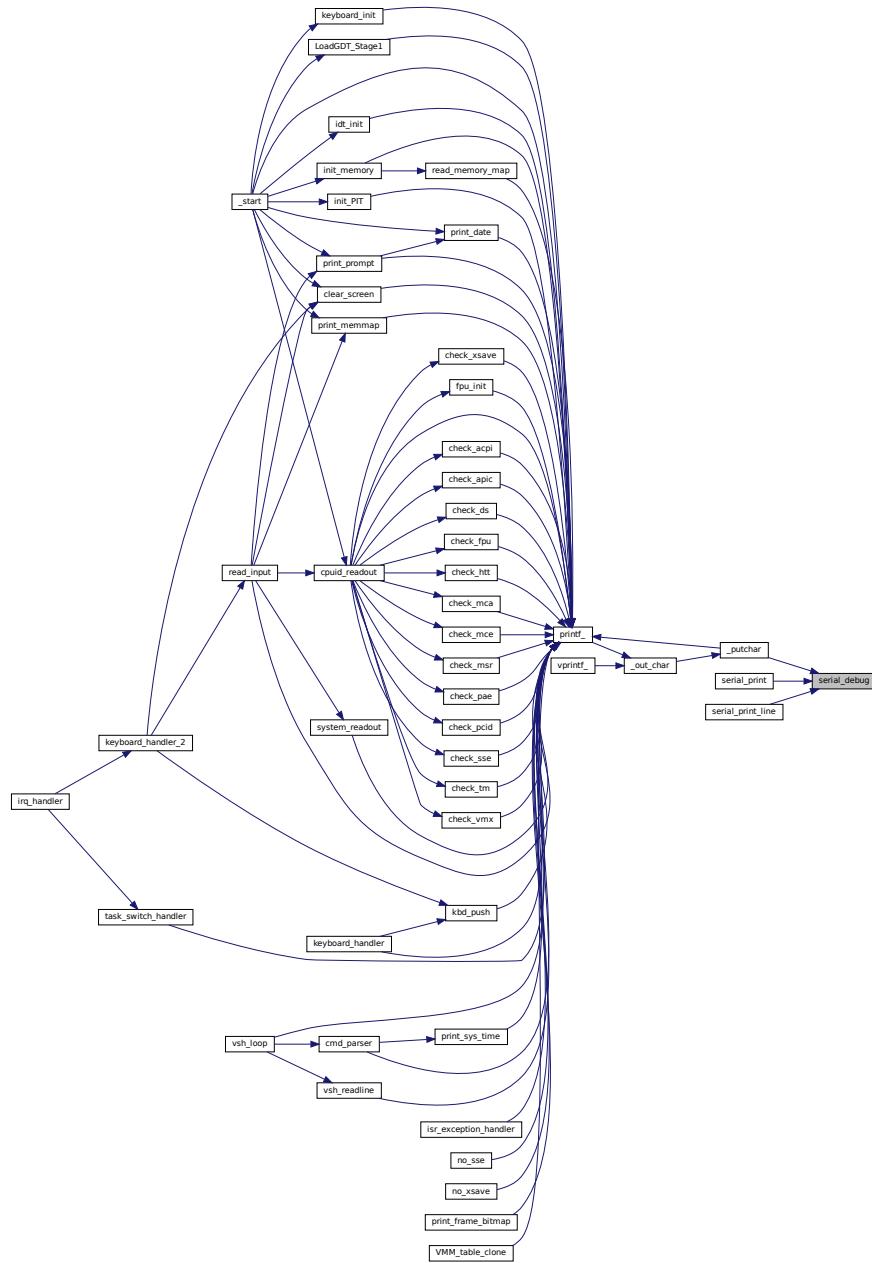
Here is the caller graph for this function:



#### 4.11.1.4 serial\_debug()

```
void serial_debug (
    int )
```

Here is the caller graph for this function:



#### 4.11.2 Variable Documentation

#### 4.11.2.1 desc

```
struct GDT_Desc desc
```

##### Initial value:

```
= {  
    .size = sizeof(gdt) - 1,  
    .offset = (uint64_t)&gdt}
```

Definition at line 19 of file [gdt.c](#).

#### 4.11.2.2 gdt

```
ALIGN_4K struct GDT gdt
```

Definition at line 19 of file [gdt.c](#).

#### 4.11.2.3 ist1Stack

```
uint8_t ist1Stack[0x100000]
```

Definition at line 12 of file [gdt.c](#).

#### 4.11.2.4 ist2Stack

```
uint8_t ist2Stack[0x100000]
```

Definition at line 13 of file [gdt.c](#).

#### 4.11.2.5 rsp0

```
uint64_t rsp0
```

Definition at line 15 of file [gdt.c](#).

#### 4.11.2.6 tss

```
struct TSS tss = {0}
```

Definition at line 15 of file [gdt.c](#).

#### 4.11.2.7 TssStack

```
uint8_t TssStack[0x100000]
```

Definition at line 11 of file [gdt.c](#).

## 4.12 gdt.c

```
00001 #include "include/gdt.h"
00002 #include "include/global_defs.h"
00003 #include <stdint.h>
00004 #include "include/printf.h"
00005 #include "include/tss.h"
00006
00007 extern void breakpoint();
00008 extern void serial_debug(int);
00009 extern void halt();
00010
00011 uint8_t TssStack[0x100000];
00012 uint8_t ist1Stack[0x100000];
00013 uint8_t ist2Stack[0x100000];
00014
00015 uint64_t rsp0;
00016
00017 struct TSS tss = {0};
00018
00019 ALIGN_4K struct GDT gdt = {
00020     {.limit_low = 0,
00021      .base_low = 0,
00022      .base_middle = 0,
00023      .access_flag = 0x00,
00024      .limit_flags = 0x00,
00025      .base_high = 0}, // null
00026
00027     {
00028         .limit_low = 0xffff,
00029         .base_low = 0,
00030         .base_middle = 0,
00031         .access_flag = GDTAccess16Code,
00032         .limit_flags = 0b00000000,
00033         .base_high = 0 // kernel 16 bit code segment
00034
00035     },
00036
00037     {
00038
00039         .limit_low = 0xffff,
00040         .base_low = 0,
00041         .base_middle = 0,
00042         .access_flag = GDTAccess16Data,
00043         .limit_flags = 0b00000000,
00044         .base_high = 0 // kernel 16 bit data segment
00045
00046     },
00047
00048     {
00049
00050         .limit_low = 0xffff,
00051         .base_low = 0,
00052         .base_middle = 0,
00053         .access_flag = GDTAccess32Code,
00054         .limit_flags = 0b11001111,
00055         .base_high = 0 // kernel 32 bit code segment
00056
00057     },
00058
00059     {
00060
00061         .limit_low = 0xffff,
00062         .base_low = 0,
00063         .base_middle = 0,
00064         .access_flag = GDTAccess32Data,
00065         .limit_flags = 0b11001111,
00066         .base_high = 0 // kernel 32 bit data segment
00067
00068     },
00069
00070     {.limit_low = 0,
00071      .base_low = 0,
00072      .base_middle = 0,
```

```

00073     .access_flag = GDTAccessKernelCode,
00074     .limit_flags = 0xA0,
00075     .base_high = 0}, // kernel 64 bit code segment
00076 {
00077     .limit_low = 0,
00078     .base_low = 0,
00079     .base_middle = 0,
00080     .access_flag = GDTAccessKernelData,
00081     .limit_flags = 0x80,
00082     .base_high = 0}, // kernel 64 bit data segment
00083 {
00084     .limit_low = 0,
00085     .base_low = 0,
00086     .base_middle = 0,
00087     .access_flag = 0x00,
00088     .limit_flags = 0x00,
00089     .base_high = 0}, // user null
00090 {
00091     .limit_low = 0,
00092     .base_low = 0,
00093     .base_middle = 0,
00094     .access_flag = GDTAccessUserData,
00095     .limit_flags = 0x80,
00096     .base_high = 0}, // user data segment
00097 {
00098     .limit_low = 0,
00099     .base_low = 0,
00100     .base_middle = 0,
00101     .access_flag = GDTAccessUserCode,
00102     .limit_flags = 0xA0,
00103     .base_high = 0}, // user code segment
00104 {
00105     .length = 104,
00106     .flags = 0b10001001,
00107 }, // TSS
00108 };
00109
00110 struct GDT_Desc desc = {
00111     .size = sizeof(gdt) - 1,
00112     .offset = (uint64_t)&gdt;
00114
00115 void LoadGDT_Stage1()
00116 {
00117     uint64_t address = (uint64_t)&tss;
00118
00119     gdt.tss = (struct TSS_Entry){
00120         .length = 104,
00121         .base_low = (uint16_t)address,
00122         .base_mid = (uint8_t)(address >> 16),
00123         .flags = 0b10001001,
00124         .base_high = (uint8_t)(address >> 24),
00125         .base_up = (uint32_t)(address >> 32),
00126     };
00127
00128     printf_("%s\n", "-----");
00129     printf_("%s\n", "| GDT INFO |");
00130     printf_("%s\n", "-----");
00131     printf_("%s\n", "GDT Offsets as follows: ");
00132     printf_("%s", "GDT NULL Segment: ");
00133     printf_("%x\n", (uint64_t)&gdt.null - (uint64_t)&gdt);
00134     printf_("%s", "GDT 16 Bit Code Segment: ");
00135     printf_("%x\n", (uint64_t)&gdt.seg_16_code - (uint64_t)&gdt);
00136     printf_("%s", "GDT 16 Bit Data Segment: ");
00137     printf_("%x\n", (uint64_t)&gdt.seg_16_data - (uint64_t)&gdt);
00138     printf_("%s", "GDT 32 Bit Code Segment: ");
00139     printf_("%x\n", (uint64_t)&gdt.seg_32_code - (uint64_t)&gdt);
00140     printf_("%s", "GDT 32 Bit Data Segment: ");
00141     printf_("%x\n", (uint64_t)&gdt.seg_32_data - (uint64_t)&gdt);
00142     printf_("%s", "GDT Kernel Code Segment: ");
00143     printf_("%x\n", (uint64_t)&gdt.kernelCS - (uint64_t)&gdt);
00144     printf_("%s", "GDT Kernel Data Segment: ");
00145     printf_("%x\n", (uint64_t)&gdt.kernelData - (uint64_t)&gdt);
00146     printf_("%s", "GDT User NULL Segment: ");
00147     printf_("%x\n", (uint64_t)&gdt.userNull - (uint64_t)&gdt);
00148     printf_("%s", "GDT User Code Segment: ");
00149     printf_("%x\n", (uint64_t)&gdt.userCode - (uint64_t)&gdt);
00150     printf_("%s", "GDT User Data Segment: ");
00151     printf_("%x\n", (uint64_t)&gdt.userData - (uint64_t)&gdt);
00152     printf_("%s", "GDT TSS Segment: ");
00153     printf_("%x\n", (uint64_t)&gdt.tss - (uint64_t)&gdt);
00154     printf_("%s\n", "-----");
00155     printf_("%s\n", "Expected GDTR Data as Follows: ");
00156     printf_("%s", "GDTR Size: ");
00157     printf_("%x\n", (uint16_t)&desc.size);
00158     printf_("%s", "GDTR Offset: ");
00159     printf_("%x\n", (uint64_t)&desc.offset);

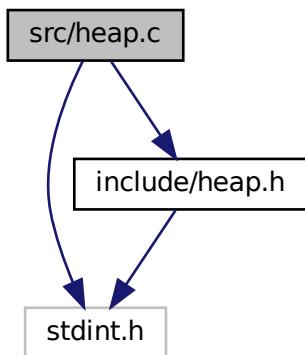
```

```

00163     printf_("%s\n", "-----");
00164     tss.rsp0 = (uint64_t)TssStack + sizeof(TssStack);
00165     tss.ist1 = (uint64_t)ist1Stack + sizeof(ist1Stack);
00166
00167     rsp0 = tss.rsp0;
00168
00169     printf_("0x%llx\n", tss.rsp0);
00170
00171     __asm__ volatile("lgdt %0"
00172                     :
00173                     : "m"(desc));
00174
00175     __asm__ volatile("push $0x28\n"
00176                     "    lea lf(%rip), %%rax\n"
00177                     "    push %%rax\n"
00178                     "    lretq\n"
00179                     "    1:\n"
00180                     :
00181                     :
00182                     : "rax", "memory");
00183
00184     __asm__ volatile("mov %0, %%ds\n"
00185                     "    mov %0, %%es\n"
00186                     "    mov %0, %%gs\n"
00187                     "    mov %0, %%fs\n"
00188                     "    mov %0, %%ss\n"
00189                     :
00190                     : "a"((uint16_t)0x30));
00191
00192     __asm__ volatile("ltr %0"
00193                     :
00194                     : "a"((uint16_t)GDTTSSSegment));
00195
00196 // breakpoint();
00197 }
```

## 4.13 src/heap.c File Reference

```
#include <stdint.h>
#include "include/heap.h"
Include dependency graph for heap.c:
```



## Functions

- void [k\\_heapBMInit](#) (KHEAPBM \*heap)
- int [k\\_heapBMAddBlock](#) (KHEAPBM \*heap, uintptr\_t addr, uint64\_t size, uint64\_t bsize)

- `uintptr_t k_heapBMGetBMSize (uintptr_t size, uint64_t bsize)`
- `int k_heapBMAddBlockEx (KHEAPBM *heap, uintptr_t addr, uint64_t size, uint64_t bsize, KHEAPBLOCKBM *b, uint8_t *bm, uint8_t isBMInside)`
- `static uint8_t k_heapBMGetNID (uint8_t a, uint8_t b)`
- `void * k_heapBMAAlloc (KHEAPBM *heap, uint64_t size)`
- `void * k_heapBMAAllocBound (KHEAPBM *heap, uint64_t size, uint64_t bound)`
- `void k_heapBMSet (KHEAPBM *heap, uintptr_t ptr, uintptr_t size, uint8_t rval)`
- `void k_heapBMFree (KHEAPBM *heap, void *ptr)`

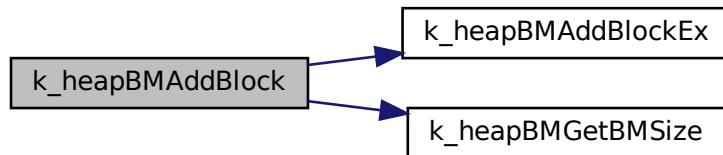
### 4.13.1 Function Documentation

#### 4.13.1.1 k\_heapBMAddBlock()

```
int k_heapBMAddBlock (
    KHEAPBM * heap,
    uintptr_t addr,
    uint64_t size,
    uint64_t bsize )
```

Definition at line 11 of file [heap.c](#).

Here is the call graph for this function:



#### 4.13.1.2 k\_heapBMAddBlockEx()

```
int k_heapBMAddBlockEx (
    KHEAPBM * heap,
    uintptr_t addr,
    uint64_t size,
    uint64_t bsize,
    KHEAPBLOCKBM * b,
    uint8_t * bm,
    uint8_t isBMInside )
```

Definition at line 29 of file [heap.c](#).

Here is the caller graph for this function:



#### 4.13.1.3 k\_heapBMAalloc()

```
void* k_heapBMAalloc (
    KHEAPBM * heap,
    uint64_t size )
```

Definition at line 77 of file [heap.c](#).

Here is the call graph for this function:

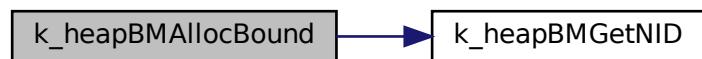


#### 4.13.1.4 k\_heapBMAallocBound()

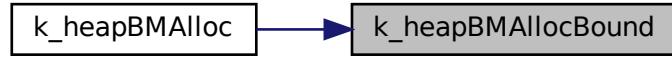
```
void* k_heapBMAallocBound (
    KHEAPBM * heap,
    uint64_t size,
    uint64_t bound )
```

Definition at line 82 of file [heap.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.13.1.5 k\_heapBMFree()

```
void k_heapBMFree (
    KHEAPBM * heap,
    void * ptr )
```

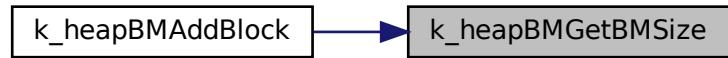
Definition at line 225 of file [heap.c](#).

#### 4.13.1.6 k\_heapBMGetBMSize()

```
uintptr_t k_heapBMGetBMSize (
    uintptr_t size,
    uint64_t bsize )
```

Definition at line 24 of file [heap.c](#).

Here is the caller graph for this function:



#### 4.13.1.7 k\_heapBMGetNID()

```
static uint8_t k_heapBMGetNID (
    uint8_t a,
    uint8_t b ) [static]
```

Definition at line 69 of file [heap.c](#).

Here is the caller graph for this function:



#### 4.13.1.8 k\_heapBMInit()

```
void k_heapBMInit (
    KHEAPBM * heap )
```

Definition at line 6 of file [heap.c](#).

#### 4.13.1.9 k\_heapBMSet()

```
void k_heapBMSet (
    KHEAPBM * heap,
    uintptr_t ptr,
    uintptr_t size,
    uint8_t rval )
```

Definition at line 154 of file [heap.c](#).

### 4.14 heap.c

```
00001 /* Leonard Kevin McGuire Jr (kmcg3413@gmail.com) (www.kmcg3413.net) */
00002
00003 #include <stdint.h>
00004 #include "include/heap.h"
00005
00006 void k_heapBMInit(KHEAPBM *heap)
00007 {
00008     heap->fblock = 0;
00009 }
00010
00011 int k_heapBMAddBlock(KHEAPBM *heap, uintptr_t addr, uint64_t size, uint64_t bsize)
00012 {
00013     KHEAPBLOCKBM *b;
00014     uintptr_t bmsz;
00015     uint8_t *bm;
00016 }
```

```

00017     b = (KHEAPBLOCKBM *)addr;
00018     bmsz = k_heapBMGetBMSize(size, bsize);
00019     bm = (uint8_t *)(addr + sizeof(KHEAPBLOCKBM));
00020     /* important to set isBMINside... (last argument) */
00021     return k_heapBMAddBlockEx(heap, addr + sizeof(KHEAPBLOCKBM), size - sizeof(KHEAPBLOCKBM), bsize,
00022     b, bm, 1);
00023
00024 uintptr_t k_heapBMGetBMSize(uintptr_t size, uint64_t bsize)
00025 {
00026     return size / bsize;
00027 }
00028
00029 int k_heapBMAddBlockEx(KHEAPBM *heap, uintptr_t addr, uint64_t size, uint64_t bsize, KHEAPBLOCKBM *b,
00030     uint8_t *bm, uint8_t isBMINside)
00031 {
00032     uint64_t bcnt;
00033     uint64_t x;
00034
00035     b->size = size;
00036     b->bsize = bsize;
00037     b->data = addr;
00038     b->bm = bm;
00039
00040     b->next = heap->fblock;
00041     heap->fblock = b;
00042
00043     bcnt = size / bsize;
00044
00045     /* clear bitmap */
00046     for (x = 0; x < bcnt; ++x)
00047     {
00048         bm[x] = 0;
00049     }
00050
00051     bcnt = (bcnt / bsize) * bsize < bcnt ? bcnt / bsize + 1 : bcnt / bsize;
00052
00053     /* if BM is not inside leave this space available */
00054     if (isBMINside)
00055     {
00056         /* reserve room for bitmap */
00057         for (x = 0; x < bcnt; ++x)
00058         {
00059             bm[x] = 5;
00060         }
00061
00062     b->lfb = bcnt - 1;
00063
00064     b->used = bcnt;
00065
00066     return 1;
00067 }
00068
00069 static uint8_t k_heapBMGetNID(uint8_t a, uint8_t b)
00070 {
00071     uint8_t c;
00072     for (c = a + 1; c == b || c == 0; ++c)
00073     ;
00074     return c;
00075 }
00076
00077 void *k_heapBMAlocate(KHEAPBM *heap, uint64_t size)
00078 {
00079     return k_heapBMAlocateBound(heap, size, 0);
00080 }
00081
00082 void *k_heapBMAlocateBound(KHEAPBM *heap, uint64_t size, uint64_t bound)
00083 {
00084     KHEAPBLOCKBM *b;
00085     uint8_t *bm;
00086     uint64_t bcnt;
00087     uint64_t x, y, z;
00088     uint64_t bneed;
00089     uint8_t nid;
00090     uint64_t max;
00091
00092     bound = ~(~0 << bound);
00093     /* iterate blocks */
00094     for (b = heap->fblock; b; b = b->next)
00095     {
00096         /* check if block has enough room */
00097         if (b->size - (b->used * b->bsize) >= size)
00098         {
00099             bcnt = b->size / b->bsize;
00100             bneed = (size / b->bsize) * b->bsize < size ? size / b->bsize + 1 : size / b->bsize;
00101             bm = (uint8_t *)b->bm;

```

```

00102
00103     for (x = (b->lfb + 1) >= bcnt ? 0 : b->lfb + 1); x != b->lfb; ++x)
00104     {
00105         /* just wrap around */
00106         if (x >= bcnt)
00107         {
00108             x = 0;
00109         }
00110
00111         /*
00112             this is used to allocate on specified boundaries larger than the block size
00113             */
00114         if (((x * b->bsize) + b->data) & bound) != 0)
00115             continue;
00116
00117         if (bm[x] == 0)
00118         {
00119             /* count free blocks */
00120             max = bcnt - x;
00121             for (y = 0; bm[x + y] == 0 && y < bneed && y < max; ++y)
00122                 ;
00123
00124             /* we have enough, now allocate them */
00125             if (y == bneed)
00126             {
00127                 /* find ID that does not match left or right */
00128                 nid = k_heapBMGetNID(bm[x - 1], bm[x + y]);
00129
00130                 /* allocate by setting id */
00131                 for (z = 0; z < y; ++z)
00132                 {
00133                     bm[x + z] = nid;
00134                 }
00135                 /* optimization */
00136                 b->lfb = (x + bneed) - 2;
00137
00138                 /* count used blocks NOT bytes */
00139                 b->used += y;
00140                 return (void *) ((x * b->bsize) + b->data);
00141             }
00142
00143             /* x will be incremented by one ONCE more in our FOR loop */
00144             x += (y - 1);
00145             continue;
00146         }
00147     }
00148 }
00149 }
00150
00151     return 0;
00152 }
00153
00154 void k_heapBMSet(KHEAPBM *heap, uintptr_t ptr, uintptr_t size, uint8_t rval)
00155 {
00156     KHEAPBLOCKBM *b;
00157     uintptr_t ptoff, endoff;
00158     uint64_t bi, x, ei;
00159     uint8_t *bm;
00160     uint8_t id;
00161     uint64_t max;
00162
00163     for (b = heap->fblock; b; b = b->next)
00164     {
00165         /* check if region effects block */
00166         if (
00167             /* head end resides inside block */
00168             (ptr >= b->data && ptr < b->data + b->size) ||
00169             /* tail end resides inside block */
00170             ((ptr + size) >= b->data && (ptr + size) < b->data + b->size) ||
00171             /* spans across but does not start or end in block */
00172             (ptr < b->data && (ptr + size) > b->data + b->size))
00173         {
00174             /* found block */
00175             if (ptr >= b->data)
00176             {
00177                 ptoff = ptr - b->data; /* get offset to get block */
00178                 /* block offset in BM */
00179                 bi = ptoff / b->bsize;
00180             }
00181             else
00182             {
00183                 /* do not start negative on bitmap */
00184                 bi = 0;
00185             }
00186
00187             /* access bitmap pointer in local variable */
00188             bm = b->bm;

```

```

00189         ptr = ptr + size;
00190         endoff = ptr - b->data;
00191
00192         /* end index inside bitmap */
00193         ei = (endoff / b->bsize) * b->bsize < endoff ? (endoff / b->bsize) + 1 : endoff /
00194         b->bsize;
00195         ++ei;
00196
00197         /* region could span past end of a block so adjust */
00198         max = b->size / b->bsize;
00199         max = ei > max ? max : ei;
00200
00201         /* set bitmap buckets */
00202         for (x = bi; x < max; ++x)
00203         {
00204             bm[x] = rval;
00205         }
00206
00207         /* update free block count */
00208         if (rval == 0)
00209         {
00210             b->used -= ei - bi;
00211         }
00212         else
00213         {
00214             b->used += ei - bi;
00215         }
00216
00217         /* do not return as region could span multiple blocks.. so check the rest */
00218     }
00219 }
00220
00221     /* this error needs to be raised or reported somehow */
00222     return;
00223 }
00224
00225 void k_heapBMFree(KHEAPBM *heap, void *ptr)
00226 {
00227     KHEAPBLOCKBM *b;
00228     uintptr_t ptoff;
00229     uint64_t bi, x;
00230     uint8_t *bm;
00231     uint8_t id;
00232     uint64_t max;
00233
00234     for (b = heap->fblock; b; b = b->next)
00235     {
00236         if ((uintptr_t)ptr > b->data && (uintptr_t)ptr < b->data + b->size)
00237         {
00238             /* found block */
00239             ptoff = (uintptr_t)ptr - b->data; /* get offset to get block */
00240             /* block offset in BM */
00241             bi = ptoff / b->bsize;
00242             /* ... */
00243             bm = b->bm;
00244             /* clear allocation */
00245             id = bm[bi];
00246
00247             max = b->size / b->bsize;
00248             for (x = bi; bm[x] == id && x < max; ++x)
00249             {
00250                 bm[x] = 0;
00251             }
00252             /* update free block count */
00253             b->used -= x - bi;
00254             return;
00255         }
00256     }
00257
00258     /* this error needs to be raised or reported somehow */
00259     return;
00260 }

```

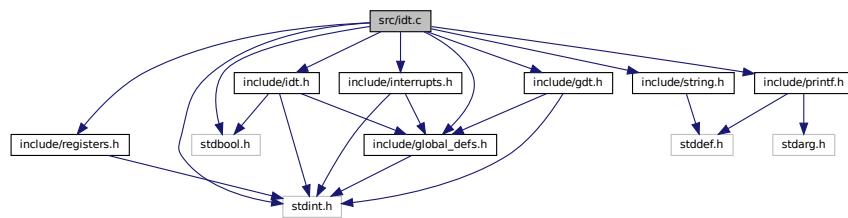
## 4.15 src/idt.c File Reference

```

#include <stdint.h>
#include "include/string.h"
#include <stdbool.h>
#include "include/global_defs.h"

```

```
#include "include/idt.h"
#include "include/printf.h"
#include "include/gdt.h"
#include "include/registers.h"
#include "include/interrupts.h"
Include dependency graph for idt.c:
```



## Functions

- void `idt_set_descriptor` (uint8\_t vector, uintptr\_t isr, uint8\_t flags, uint8\_t ist)
- void `idt_init` ()
- uint8\_t `idt_allocate_vector` ()
- void `idt_free_vector` (uint8\_t vector)

## Variables

- static ALIGN\_16BIT `idt_desc_t idt [IDT_MAX_DESCRIPTOR]`
- static `idtr_t idtr`
- static bool `vectors [IDT_MAX_DESCRIPTOR]`
- `uint64_t isr_stub_table []`

### 4.15.1 Function Documentation

#### 4.15.1.1 `idt_allocate_vector()`

```
uint8_t idt_allocate_vector (
    void )
```

Definition at line 63 of file `idt.c`.

### 4.15.1.2 idt\_free\_vector()

```
void idt_free_vector (
    uint8_t vector )
```

Definition at line 77 of file [idt.c](#).

Here is the call graph for this function:

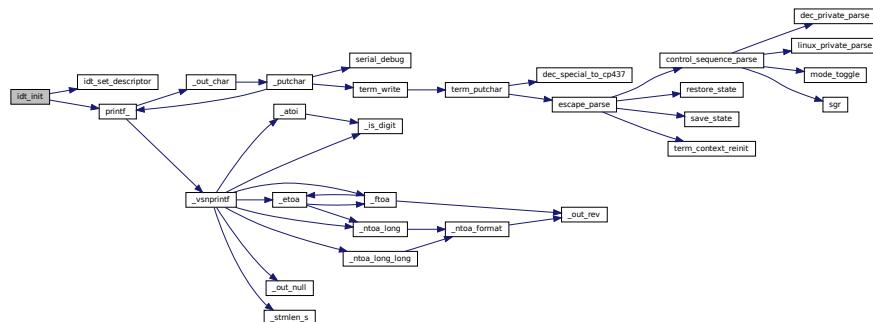


### 4.15.1.3 idt\_init()

```
void idt_init (
    void )
```

Definition at line 33 of file [idt.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

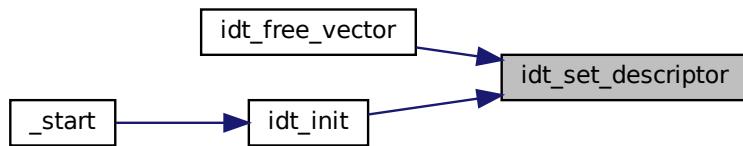


#### 4.15.1.4 idt\_set\_descriptor()

```
void idt_set_descriptor (
    uint8_t vector,
    uintptr_t isr,
    uint8_t flags,
    uint8_t ist )
```

Definition at line 20 of file [idt.c](#).

Here is the caller graph for this function:



## 4.15.2 Variable Documentation

### 4.15.2.1 idt

```
ALIGN_16BIT idt_desc_t idt[IDT_MAX_DESCRIPTOR] [static]
```

Definition at line 12 of file [idt.c](#).

### 4.15.2.2 idtr

```
idtr_t idtr [static]
```

Definition at line 14 of file [idt.c](#).

### 4.15.2.3 isr\_stub\_table

```
uint64_t isr_stub_table[] [extern]
```

#### 4.15.2.4 vectors

```
bool vectors[IDT_MAX_DESCRIPTORS] [static]
```

Definition at line 16 of file [idt.c](#).

## 4.16 idt.c

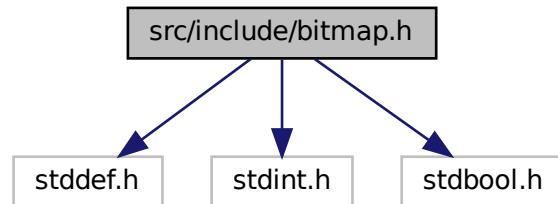
```
00001 #include <stdint.h>
00002 #include "include/string.h"
00003 #include <stdbool.h>
00004 #include "include/global_defs.h"
00005 #include "include/idt.h"
00006 #include "include/printf.h"
00007 #include "include/gdt.h"
00008 #include "include/registers.h"
00009 #include "include/interrupts.h"
00010
00011 static ALIGN_16BIT
00012     idt_desc_t idt[IDT_MAX_DESCRIPTORS];
00013
00014 static idtr_t idtr;
00015
00016 static bool vectors[IDT_MAX_DESCRIPTORS];
00017
00018 extern uint64_t isr_stub_table[];
00019
00020 void idt_set_descriptor(uint8_t vector, uintptr_t isr, uint8_t flags, uint8_t ist)
00021 {
00022     idt_desc_t *descriptor = &idt[vector];
00023
00024     descriptor->base_low = isr & 0xFFFF;
00025     descriptor->cs = GDTKernelBaseSelector;
00026     descriptor->ist = ist;
00027     descriptor->attributes = flags;
00028     descriptor->base_mid = (isr >> 16) & 0xFFFF;
00029     descriptor->base_high = (isr >> 32) & 0xFFFFFFFF;
00030     descriptor->rsv0 = 0;
00031 }
00032
00033 void idt_init()
00034 {
00035     idtr.base = (uintptr_t)&idt[0];
00036     idtr.limit = (uint16_t)sizeof(idt_desc_t) * IDT_MAX_DESCRIPTORS - 1;
00037
00038     for (uint8_t vector = 0; vector < IDT_CPU_EXCEPTION_COUNT + IDT_HDW_INTERRUPT_COUNT; vector++)
00039     {
00040         if (vector >= 32)
00041         {
00042             idt_set_descriptor(vector, isr_stub_table[vector], IDT_DESCRIPTOR_EXTERNAL, 001);
00043             vectors[vector] = true;
00044
00045             printf_(""%i\n", vector);
00046         }
00047         else
00048         {
00049             idt_set_descriptor(vector, isr_stub_table[vector], IDT_DESCRIPTOR_EXCEPTION, 001);
00050             vectors[vector] = true;
00051         }
00052     }
00053
00054     __asm__ volatile("lidt %0"
00055     :
00056     : "m"(idtr)); // load the new IDT
00057     __asm__ volatile("sti"); // set the interrupt flag
00058 }
00059
00060 uint8_t idt_allocate_vector()
00061 {
00062     for (unsigned int i = 0; i < IDT_MAX_DESCRIPTORS; i++)
00063     {
00064         if (!vectors[i])
00065         {
00066             vectors[i] = true;
00067             return (uint8_t)i;
00068         }
00069     }
00070 }
```

```

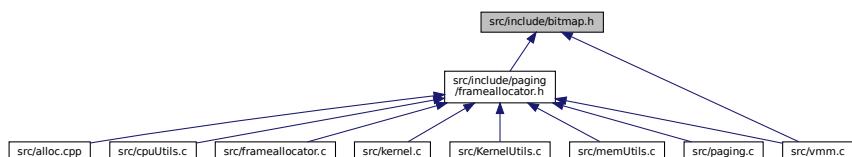
00073
00074     return 0;
00075 }
00076
00077 void idt_free_vector(uint8_t vector)
00078 {
00079     idt_set_descriptor(vector, 0, 0, 0);
00080     vectors[vector] = false;
00081 }
```

## 4.17 src/include(bitmap.h File Reference

```
#include <stddef.h>
#include <stdint.h>
#include <stdbool.h>
Include dependency graph for bitmap.h:
```



This graph shows which files directly or indirectly include this file:



## Functions

- bool [bitmap\\_get](#) (uint8\_t \*bitmap, size\_t bit)
- void [bitmap\\_set](#) (uint8\_t \*bitmap, size\_t bit, uint8\_t value)

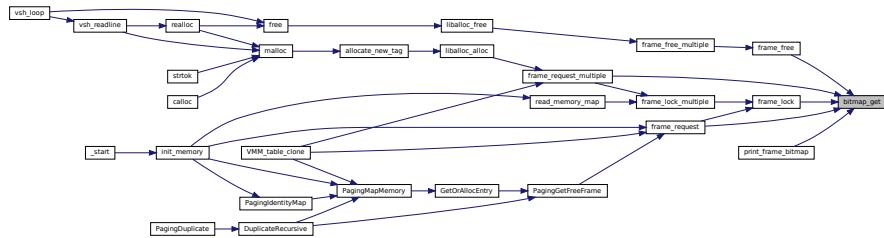
### 4.17.1 Function Documentation

### 4.17.1.1 bitmap\_get()

```
bool bitmap_get (
    uint8_t * bitmap,
    size_t bit ) [inline]
```

Definition at line 6 of file [bitmap.h](#).

Here is the caller graph for this function:

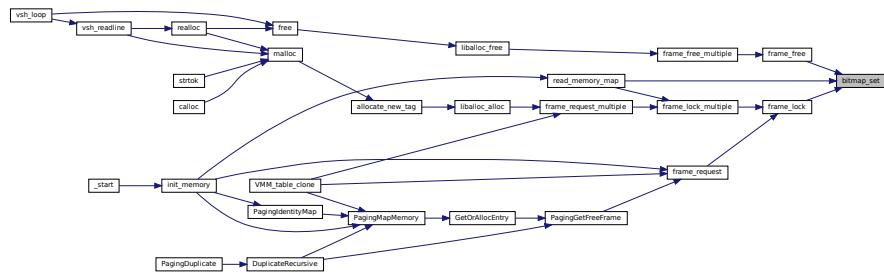


### 4.17.1.2 bitmap\_set()

```
void bitmap_set (
    uint8_t * bitmap,
    size_t bit,
    uint8_t value ) [inline]
```

Definition at line 15 of file [bitmap.h](#).

Here is the caller graph for this function:



## 4.18 bitmap.h

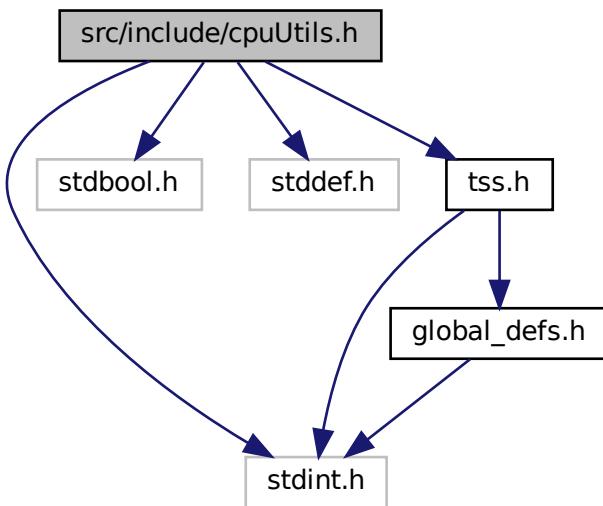
```

00001 #pragma once
00002 #include <stddef.h>
00003 #include <stdint.h>
00004 #include <stdbool.h>
00005
00006 inline bool bitmap_get(uint8_t *bitmap, size_t bit)
00007 {
00008     uint64_t byteIndex = bit / 8;
00009     uint8_t bitIndex = bit % 8;
00010     uint8_t bitIndexer = 0b10000000 » bitIndex;
00011
00012     return (bitmap[byteIndex] & bitIndexer) > 0;
00013 }
00014
00015 inline void bitmap_set(uint8_t *bitmap, size_t bit, uint8_t value)
00016 {
00017     uint64_t byteIndex = bit / 8;
00018     uint8_t bitIndex = bit % 8;
00019     uint8_t bitIndexer = 0b10000000 » bitIndex;
00020
00021     bitmap[byteIndex] &= ~bitIndexer;
00022
00023     if (value)
00024     {
00025         bitmap[byteIndex] |= bitIndexer;
00026     }
00027 }
```

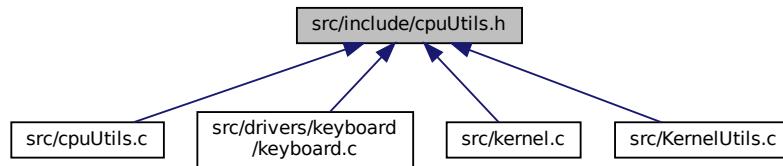
## 4.19 src/include/cpuUtils.h File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>
#include "tss.h"

Include dependency graph for cpuUtils.h:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `cpu_ctx`
- struct `cpu_local`

## Macros

- #define \_CPU\_UTILS\_H
- #define CPUID\_VENDOR\_OLDAMD "AMDisbetter!"
- #define CPUID\_VENDOR\_AMD "AuthenticAMD"
- #define CPUID\_VENDOR\_INTEL "GenuineIntel"
- #define CPUID\_VENDOR\_VIA "VIA VIA VIA "
- #define CPUID\_VENDOR\_OLDTRANSMETA "TransmetaCPU"
- #define CPUID\_VENDOR\_TRANSMETA "GenuineTMx86"
- #define CPUID\_VENDOR\_CYRIX "CyrixInstead"
- #define CPUID\_VENDOR\_CENTAUR "CentaurHauls"
- #define CPUID\_VENDOR\_NEXGEN "NexGenDriven"
- #define CPUID\_VENDOR\_UMC "UMC UMC UMC "
- #define CPUID\_VENDOR\_SIS "SiS SiS SiS "
- #define CPUID\_VENDOR\_NSC "Geode by NSC"
- #define CPUID\_VENDOR\_RISE "RiseRiseRise"
- #define CPUID\_VENDOR\_VORTEX "Vortex86 SoC"
- #define CPUID\_VENDOR\_OLDAO486 "GenuineAO486"
- #define CPUID\_VENDOR\_AO486 "MiSTer AO486"
- #define CPUID\_VENDOR\_ZHAOXIN "Shanghai "
- #define CPUID\_VENDOR\_HYGON "HygonGenuine"
- #define CPUID\_VENDOR\_ELBRUS "E2K MACHINE "
- #define CPUID\_VENDOR\_QEMU "TCGTCGTCGTCG"
- #define CPUID\_VENDOR\_KVM "KVMKVMKVM "
- #define CPUID\_VENDOR\_VMWWARE "VMwareVMware"
- #define CPUID\_VENDOR\_VIRTUALBOX "VBoxVBoxVBox"
- #define CPUID\_VENDOR\_XEN "XenVMMXenVMM"
- #define CPUID\_VENDOR\_HYPERV "Microsoft Hv"
- #define CPUID\_VENDOR\_PARALLELS "prl hyperv "
- #define CPUID\_VENDOR\_PARALLELS\_ALT "lrpepyh vr "
- #define CPUID\_VENDOR\_BHYVE "bhyve bhyve "
- #define CPUID\_VENDOR\_QNX "QNXQVMBSQG "

## Functions

- void `cpu_init` (void)
- static void `xsave` (void \*ctx)
- static void `xrstor` (void \*ctx)
- static void `fxsave` (void \*ctx)
- static void `fxrstor` (void \*ctx)
- static bool `interrupt_state` (void)
- static uint64\_t `rdmsr` (uint32\_t msr)
- static uint64\_t `wrmsr` (uint32\_t msr, uint64\_t val)
- static void `set_kernel_gs_base` (void \*addr)
- static void `set_gs_base` (void \*addr)
- static void `set_fs_base` (void \*addr)
- static void \* `get_kernel_gs_base` (void)
- static void \* `get_gs_base` (void)
- static void \* `get_fs_base` (void)
- struct `cpu_local` \* `this_cpu` (void)
- int `get_model` (void)
 

*Get the CPUID model.*
- void `cpuid_readout` ()
 

*Print a message to the standard output. This is called from cpuid\_read ()*
- void `detect_cpu` (void)

## Variables

- bool `sysenter`
- char `CPU_vendor` []
- size\_t `fpu_bank_size`
- void(\* `fpu_save` )(void \*ctx)
- void(\* `fpu_rest` )(void \*ctx)

### 4.19.1 Data Structure Documentation

#### 4.19.1.1 struct `cpu_ctx`

Definition at line 49 of file `cpuUtils.h`.

##### Data Fields

<code>uint64_t</code>	<code>cs</code>	
<code>uint64_t</code>	<code>ds</code>	
<code>uint64_t</code>	<code>err</code>	
<code>uint64_t</code>	<code>es</code>	
<code>uint64_t</code>	<code>r10</code>	
<code>uint64_t</code>	<code>r11</code>	
<code>uint64_t</code>	<code>r12</code>	
<code>uint64_t</code>	<code>r13</code>	
<code>uint64_t</code>	<code>r14</code>	
<code>uint64_t</code>	<code>r15</code>	
<code>uint64_t</code>	<code>r8</code>	
<code>uint64_t</code>	<code>r9</code>	

**Data Fields**

uint64_t	rax	
uint64_t	rbp	
uint64_t	rbx	
uint64_t	rcx	
uint64_t	rdi	
uint64_t	rdx	
uint64_t	rflags	
uint64_t	rip	
uint64_t	rsi	
uint64_t	rsp	
uint64_t	ss	

**4.19.2 Macro Definition Documentation****4.19.2.1 \_CPU\_UTILS\_H**

```
#define _CPU_UTILS_H
```

Definition at line 3 of file [cpuUtils.h](#).

**4.19.2.2 CPUID\_VENDOR\_AMD**

```
#define CPUID_VENDOR_AMD "AuthenticAMD"
```

Definition at line 12 of file [cpuUtils.h](#).

**4.19.2.3 CPUID\_VENDOR\_AO486**

```
#define CPUID_VENDOR_AO486 "MiSTer AO486"
```

Definition at line 26 of file [cpuUtils.h](#).

**4.19.2.4 CPUID\_VENDOR\_BHYVE**

```
#define CPUID_VENDOR_BHYVE "bhyve bhyve "
```

Definition at line 40 of file [cpuUtils.h](#).

#### 4.19.2.5 CPUID\_VENDOR\_CENTAUR

```
#define CPUID_VENDOR_CENTAUR "CentaurHauls"
```

Definition at line 18 of file [cpuUtils.h](#).

#### 4.19.2.6 CPUID\_VENDOR\_CYRIX

```
#define CPUID_VENDOR_CYRIX "CyrixInstead"
```

Definition at line 17 of file [cpuUtils.h](#).

#### 4.19.2.7 CPUID\_VENDOR\_ELBRUS

```
#define CPUID_VENDOR_ELBRUS "E2K MACHINE "
```

Definition at line 29 of file [cpuUtils.h](#).

#### 4.19.2.8 CPUID\_VENDOR\_HYGON

```
#define CPUID_VENDOR_HYGON "HygonGenuine"
```

Definition at line 28 of file [cpuUtils.h](#).

#### 4.19.2.9 CPUID\_VENDOR\_HYPERV

```
#define CPUID_VENDOR_HYPERV "Microsoft Hv"
```

Definition at line 37 of file [cpuUtils.h](#).

#### 4.19.2.10 CPUID\_VENDOR\_INTEL

```
#define CPUID_VENDOR_INTEL "GenuineIntel"
```

Definition at line 13 of file [cpuUtils.h](#).

#### 4.19.2.11 CPUID\_VENDOR\_KVM

```
#define CPUID_VENDOR_KVM "KVMKVMKVM "
```

Definition at line 33 of file [cpuUtils.h](#).

#### 4.19.2.12 CPUID\_VENDOR\_NEXGEN

```
#define CPUID_VENDOR_NEXGEN "NexGenDriven"
```

Definition at line 19 of file [cpuUtils.h](#).

#### 4.19.2.13 CPUID\_VENDOR\_NSC

```
#define CPUID_VENDOR_NSC "Geode by NSC"
```

Definition at line 22 of file [cpuUtils.h](#).

#### 4.19.2.14 CPUID\_VENDOR\_OLDAMD

```
#define CPUID_VENDOR_OLDAMD "AMDisbetter!"
```

Definition at line 11 of file [cpuUtils.h](#).

#### 4.19.2.15 CPUID\_VENDOR\_OLDAO486

```
#define CPUID_VENDOR_OLDAO486 "GenuineAO486"
```

Definition at line 25 of file [cpuUtils.h](#).

#### 4.19.2.16 CPUID\_VENDOR\_OLDTRANSMETA

```
#define CPUID_VENDOR_OLDTRANSMETA "TransmetaCPU"
```

Definition at line 15 of file [cpuUtils.h](#).

#### 4.19.2.17 CPUID\_VENDOR\_PARALLELS

```
#define CPUID_VENDOR_PARALLELS " prl hyperv "
```

Definition at line 38 of file [cpuUtils.h](#).

#### 4.19.2.18 CPUID\_VENDOR\_PARALLELS\_ALT

```
#define CPUID_VENDOR_PARALLELS_ALT " lrpepyh vr "
```

Definition at line 39 of file [cpuUtils.h](#).

#### 4.19.2.19 CPUID\_VENDOR\_QEMU

```
#define CPUID_VENDOR_QEMU "TCGTCGTCGTCG"
```

Definition at line 32 of file [cpuUtils.h](#).

#### 4.19.2.20 CPUID\_VENDOR\_QNX

```
#define CPUID_VENDOR_QNX " QNXQVMBSQG "
```

Definition at line 41 of file [cpuUtils.h](#).

#### 4.19.2.21 CPUID\_VENDOR\_RISE

```
#define CPUID_VENDOR_RISE "RiseRiseRise"
```

Definition at line 23 of file [cpuUtils.h](#).

#### 4.19.2.22 CPUID\_VENDOR\_SIS

```
#define CPUID_VENDOR_SIS "Sis Sis Sis Sis "
```

Definition at line 21 of file [cpuUtils.h](#).

#### 4.19.2.23 CPUID\_VENDOR\_TRANSMETA

```
#define CPUID_VENDOR_TRANSMETA "GenuineTMx86"
```

Definition at line 16 of file [cpuUtils.h](#).

#### 4.19.2.24 CPUID\_VENDOR\_UMC

```
#define CPUID_VENDOR_UMC "UMC UMC UMC "
```

Definition at line 20 of file [cpuUtils.h](#).

#### 4.19.2.25 CPUID\_VENDOR\_VIA

```
#define CPUID_VENDOR_VIA "VIA VIA VIA "
```

Definition at line 14 of file [cpuUtils.h](#).

#### 4.19.2.26 CPUID\_VENDOR\_VIRTUALBOX

```
#define CPUID_VENDOR_VIRTUALBOX "VBoxVBoxVBox"
```

Definition at line 35 of file [cpuUtils.h](#).

#### 4.19.2.27 CPUID\_VENDOR\_VMWARE

```
#define CPUID_VENDOR_VMWARE "VMwareVMware"
```

Definition at line 34 of file [cpuUtils.h](#).

#### 4.19.2.28 CPUID\_VENDOR\_VORTEX

```
#define CPUID_VENDOR_VORTEX "Vortex86 SoC"
```

Definition at line 24 of file [cpuUtils.h](#).

#### 4.19.2.29 CPUID\_VENDOR\_XEN

```
#define CPUID_VENDOR_XEN "XenVMMXenVMM"
```

Definition at line [36](#) of file [cpuUtils.h](#).

#### 4.19.2.30 CPUID\_VENDOR\_ZHAOXIN

```
#define CPUID_VENDOR_ZHAOXIN " Shanghai "
```

Definition at line [27](#) of file [cpuUtils.h](#).

### 4.19.3 Function Documentation

#### 4.19.3.1 cpu\_init()

```
void cpu_init (
    void  )
```

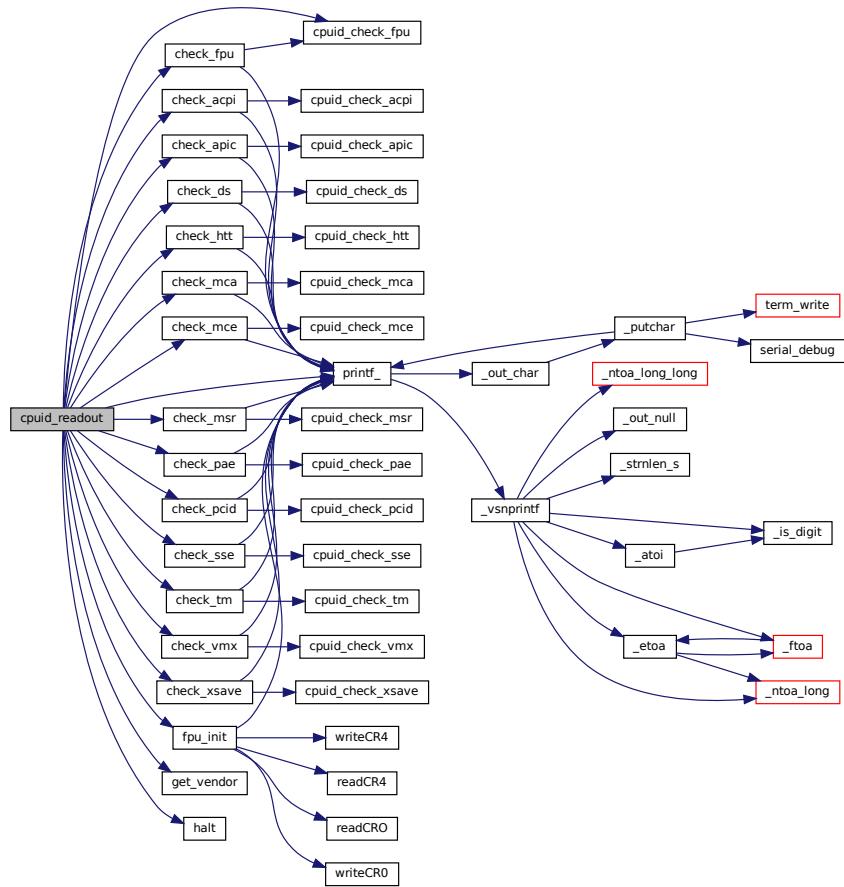
#### 4.19.3.2 cpuid\_readout()

```
void cpuid_readout ( )
```

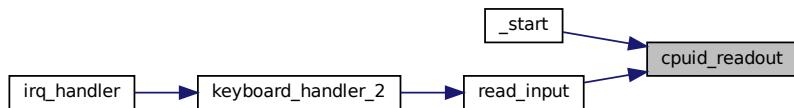
Print a message to the standard output. This is called from cpuid\_read ()

Definition at line [44](#) of file [cpuUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

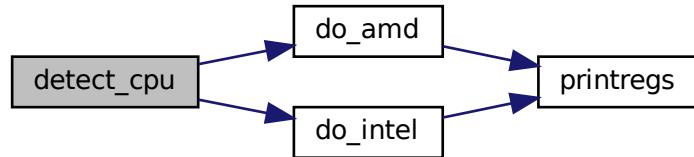


#### 4.19.3.3 detect\_cpu()

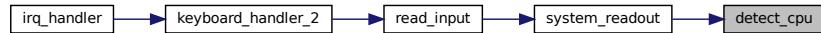
```
void detect_cpu (
    void )
```

Definition at line 42 of file [cpudet-clean.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.19.3.4 fxrstor()

```
static void fxrstor (
    void * ctx ) [inline], [static]
```

Definition at line 121 of file [cpuUtils.h](#).

#### 4.19.3.5 fxsave()

```
static void fxsave (
    void * ctx ) [inline], [static]
```

Definition at line 112 of file [cpuUtils.h](#).

#### 4.19.3.6 `get_fs_base()`

```
static void* get_fs_base (
    void ) [inline], [static]
```

Definition at line 186 of file [cpuUtils.h](#).

Here is the call graph for this function:

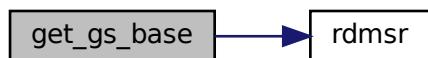


#### 4.19.3.7 `get_gs_base()`

```
static void* get_gs_base (
    void ) [inline], [static]
```

Definition at line 181 of file [cpuUtils.h](#).

Here is the call graph for this function:



#### 4.19.3.8 `get_kernel_gs_base()`

```
static void* get_kernel_gs_base (
    void ) [inline], [static]
```

Definition at line 176 of file [cpuUtils.h](#).

Here is the call graph for this function:



#### 4.19.3.9 get\_model()

```
int get_model (
    void )
```

Get the CPUID model.

##### Returns

The number of EBX

Definition at line 101 of file [cpuUtils.c](#).

#### 4.19.3.10 interrupt\_state()

```
static bool interrupt_state (
    void ) [inline], [static]
```

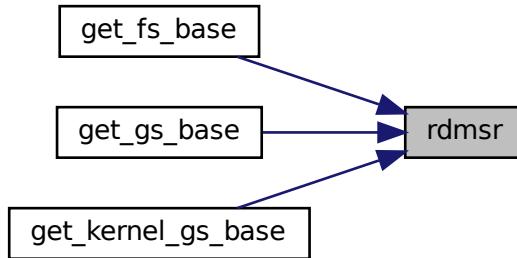
Definition at line 130 of file [cpuUtils.h](#).

#### 4.19.3.11 rdmsr()

```
static uint64_t rdmsr (
    uint32_t msr ) [inline], [static]
```

Definition at line 138 of file [cpuUtils.h](#).

Here is the caller graph for this function:

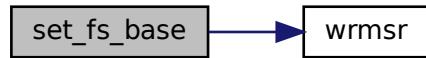


#### 4.19.3.12 set\_fs\_base()

```
static void set_fs_base (
    void * addr ) [inline], [static]
```

Definition at line 171 of file [cpuUtils.h](#).

Here is the call graph for this function:

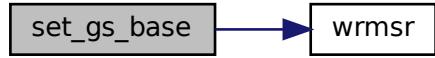


#### 4.19.3.13 set\_gs\_base()

```
static void set_gs_base (
    void * addr ) [inline], [static]
```

Definition at line 166 of file [cpuUtils.h](#).

Here is the call graph for this function:



#### 4.19.3.14 set\_kernel\_gs\_base()

```
static void set_kernel_gs_base (
    void * addr ) [inline], [static]
```

Definition at line 161 of file [cpuUtils.h](#).

Here is the call graph for this function:



#### 4.19.3.15 this\_cpu()

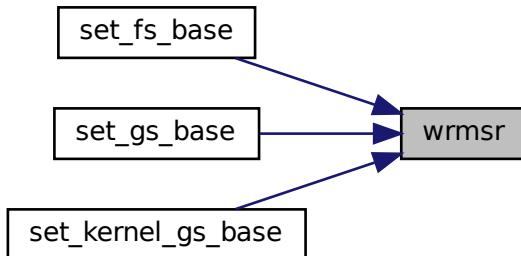
```
struct cpu_local* this_cpu (
    void )
```

#### 4.19.3.16 wrmsr()

```
static uint64_t wrmsr (
    uint32_t msr,
    uint64_t val ) [inline], [static]
```

Definition at line 149 of file [cpuUtils.h](#).

Here is the caller graph for this function:



#### 4.19.3.17 xrstor()

```
static void xrstor (
    void * ctx ) [inline], [static]
```

Definition at line 103 of file [cpuUtils.h](#).

#### 4.19.3.18 xsave()

```
static void xsave (
    void * ctx ) [inline], [static]
```

Definition at line 94 of file [cpuUtils.h](#).

### 4.19.4 Variable Documentation

#### 4.19.4.1 CPU\_vendor

```
char CPU_vendor[ ] [extern]
```

Definition at line 37 of file [cpuUtils.c](#).

#### 4.19.4.2 fpu\_bank\_size

```
size_t fpu_bank_size [extern]
```

#### 4.19.4.3 fpu\_rest

```
void(* fpu_rest) (void *ctx) (
    void * ctx ) [extern]
```

#### 4.19.4.4 fpu\_save

```
void(* fpu_save) (void *ctx) (
    void * ctx ) [extern]
```

#### 4.19.4.5 sysenter

```
bool sysenter [extern]
```

### 4.20 cpuUtils.h

```
00001 #pragma once
00002 #ifndef _CPU_UTILS_H
00003 #define _CPU_UTILS_H
00004
00005 #include <stdint.h>
00006 #include <stdbool.h>
00007 #include <stddef.h>
00008 #include "tss.h"
00009
00010 /* Vendor strings from CPUs. */
00011 #define CPUID_VENDOR_OLDAMD "AMDisbetter!" // Early engineering samples of AMD K5 processor
00012 #define CPUID_VENDOR_AMD "AuthenticAMD"
00013 #define CPUID_VENDOR_INTEL "GenuineIntel"
00014 #define CPUID_VENDOR_VIA "VIA VIA VIA"
00015 #define CPUID_VENDOR_OLDTRANSMETA "TransmetaCPU"
00016 #define CPUID_VENDOR_TRANSMETA "GenuineTMx86"
00017 #define CPUID_VENDOR_CYRIX "CyrixInstead"
00018 #define CPUID_VENDOR_CENTAUR "CentaurHauls"
00019 #define CPUID_VENDOR_NEXGEN "NexGenDriven"
00020 #define CPUID_VENDOR_UMC "UMC UMC UMC"
00021 #define CPUID_VENDOR_SIS "SIS Sis Sis"
00022 #define CPUID_VENDOR_NSC "Geode by NSC"
00023 #define CPUID_VENDOR_RISE "RiseRiseRise"
```

```
00024 #define CPUID_VENDOR_VORTEX "Vortex86 SoC"
00025 #define CPUID_VENDOR_OLDAO486 "GenuineAO486"
00026 #define CPUID_VENDOR_AO486 "MiSTer AO486"
00027 #define CPUID_VENDOR_ZHAOXIN " Shanghai "
00028 #define CPUID_VENDOR_HYGON "HygonGenuine"
00029 #define CPUID_VENDOR_ELBRUS "E2K MACHINE "
00030
00031 /* Vendor strings from hypervisors. */
00032 #define CPUID_VENDOR_QEMU "TCGTCGTCGTCG"
00033 #define CPUID_VENDOR_KVM " KVMKVMKVM "
00034 #define CPUID_VENDOR_VMWARE "VMwareVMware"
00035 #define CPUID_VENDOR_VIRTUALBOX "VBoxVBoxVBox"
00036 #define CPUID_VENDOR_XEN "XenVMMXenVMM"
00037 #define CPUID_VENDOR_HYPERV "Microsoft Hv"
00038 #define CPUID_VENDOR_PARALLELS " prl hyperv "
00039 #define CPUID_VENDOR_PARALLELS_ALT " lrpepyh vr " // Sometimes Parallels incorrectly encodes "prl
hyperv" as "lrpepyh vr" due to an endianness mismatch.
00040 #define CPUID_VENDOR_BHYVE "bhyve bhyve "
00041 #define CPUID_VENDOR_QNX " QNXQVMBSQG "
00042
00043 extern bool sysenter;
00044
00045 extern char CPU_vendor[];
00046
00047 struct thread;
00048
00049 struct cpu_ctx
00050 {
00051     uint64_t ds;
00052     uint64_t es;
00053     uint64_t rax;
00054     uint64_t rbx;
00055     uint64_t rcx;
00056     uint64_t rdx;
00057     uint64_t rsi;
00058     uint64_t rdi;
00059     uint64_t rbp;
00060     uint64_t r8;
00061     uint64_t r9;
00062     uint64_t r10;
00063     uint64_t r11;
00064     uint64_t r12;
00065     uint64_t r13;
00066     uint64_t r14;
00067     uint64_t r15;
00068     uint64_t err;
00069     uint64_t rip;
00070     uint64_t cs;
00071     uint64_t rflags;
00072     uint64_t rsp;
00073     uint64_t ss;
00074 };
00075
00076 struct cpu_local
00077 {
00078     int cpu_number;
00079     bool bsp;
00080     bool active;
00081     int last_run_queue_index;
00082     struct TSS tss;
00083     struct thread *idle_thread;
00084     void (*timer_function)(int, struct cpu_ctx *);
00085
00086 };
00087
00088 void cpu_init(void);
00089
00090 extern size_t fpu_bank_size;
00091 extern void (*fpu_save)(void *ctx);
00092 extern void (*fpu_rest)(void *ctx);
00093
00094 static inline void xsave(void *ctx)
00095 {
00096     asm volatile(
00097         "xsave (%0)"
00098         :
00099         : "r"(ctx), "a"(0xffffffff), "d"(0xffffffff)
00100         : "memory");
00101 }
00102
00103 static inline void xrstor(void *ctx)
00104 {
00105     asm volatile(
00106         "xrstor (%0)"
00107         :
00108         : "r"(ctx), "a"(0xffffffff), "d"(0xffffffff)
00109         : "memory");

```

```

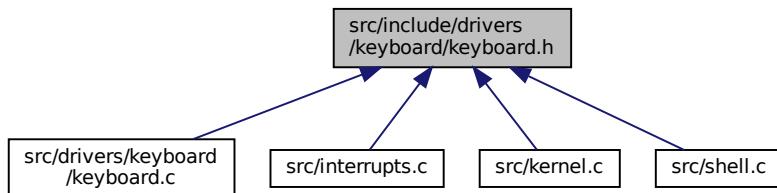
00110 }
00111
00112 static inline void fxsave(void *ctx)
00113 {
00114     asm volatile(
00115         "fxsave (%0)"
00116         :
00117         : "r"(ctx)
00118         : "memory");
00119 }
00120
00121 static inline void fxrstor(void *ctx)
00122 {
00123     asm volatile(
00124         "fxrstor (%0)"
00125         :
00126         : "r"(ctx)
00127         : "memory");
00128 }
00129
00130 static inline bool interrupt_state(void)
00131 {
00132     uint64_t flags;
00133     asm volatile("pushfq; pop %0"
00134                 : "=rm"(flags));
00135     return flags & (1 << 9);
00136 }
00137
00138 static inline uint64_t rdmsr(uint32_t msr)
00139 {
00140     uint32_t edx = 0, eax = 0;
00141     asm volatile(
00142         "rdmsr\n\t"
00143         : "=a"(eax), "=d"(edx)
00144         : "c"(msr)
00145         : "memory");
00146     return ((uint64_t)edx << 32) | eax;
00147 }
00148
00149 static inline uint64_t wrmsr(uint32_t msr, uint64_t val)
00150 {
00151     uint32_t eax = (uint32_t)val;
00152     uint32_t edx = (uint32_t)(val >> 32);
00153     asm volatile(
00154         "wrmsr\n\t"
00155         :
00156         : "a"(eax), "d"(edx), "c"(msr)
00157         : "memory");
00158     return ((uint64_t)edx << 32) | eax;
00159 }
00160
00161 static inline void set_kernel_gs_base(void *addr)
00162 {
00163     wrmsr(0xc0000102, (uint64_t)addr);
00164 }
00165
00166 static inline void set_gs_base(void *addr)
00167 {
00168     wrmsr(0xc0000101, (uint64_t)addr);
00169 }
00170
00171 static inline void set_fs_base(void *addr)
00172 {
00173     wrmsr(0xc0000100, (uint64_t)addr);
00174 }
00175
00176 static inline void *get_kernel_gs_base(void)
00177 {
00178     return (void *)rdmsr(0xc0000102);
00179 }
00180
00181 static inline void *get_gs_base(void)
00182 {
00183     return (void *)rdmsr(0xc0000101);
00184 }
00185
00186 static inline void *get_fs_base(void)
00187 {
00188     return (void *)rdmsr(0xc0000100);
00189 }
00190
00191 struct cpu_local *this_cpu(void);
00192
00193 int get_model(void);
00194 void cpuid_readout();
00195 void detect_cpu(void);
00196

```

```
00197 #endif
```

## 4.21 src/include/drivers/keyboard/keyboard.h File Reference

This graph shows which files directly or indirectly include this file:



### Macros

- `#define KEYBOARD_DATA_PORT 0x60`
- `#define KEYBOARD_STATUS_PORT 0x64`

### Functions

- `void keyboard_handler ()`  
*This is the keyboard interrupt handler. It reads the status and pushes the keycode to the keyboard stack.*
- `void keyboard_init ()`  
*Initialize the keyboard. Unmask IRQ and read keyboard data.*
- `char k_getchar ()`  
*Get a character from k\_char.*
- `char kbd_pop ()`  
*Pop a character from the kbd stack.*
- `void keyboard_handler_2 ()`  
*Keyboard handler for 2. 0 and later.*
- `void read_input ()`  
*Read input from the keyboard.*

#### 4.21.1 Macro Definition Documentation

##### 4.21.1.1 KEYBOARD\_DATA\_PORT

```
#define KEYBOARD_DATA_PORT 0x60
```

Definition at line 3 of file [keyboard.h](#).

#### 4.21.1.2 KEYBOARD\_STATUS\_PORT

```
#define KEYBOARD_STATUS_PORT 0x64
```

Definition at line 4 of file [keyboard.h](#).

### 4.21.2 Function Documentation

#### 4.21.2.1 k\_getchar()

```
char k_getchar ( )
```

Get a character from k\_char.

##### Returns

The character or 0 if none

Definition at line 224 of file [keyboard.c](#).

#### 4.21.2.2 kbd\_pop()

```
char kbd_pop ( )
```

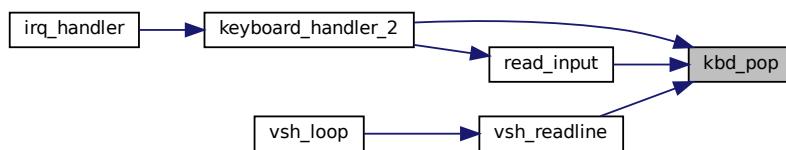
Pop a character from the kbd stack.

##### Returns

character that was popped

Definition at line 52 of file [keyboard.c](#).

Here is the caller graph for this function:



### 4.21.2.3 keyboard\_handler()

```
void keyboard_handler ( )
```

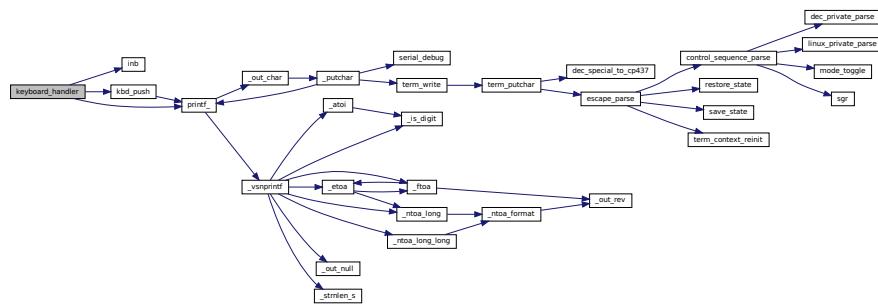
This is the keyboard interrupt handler. It reads the status and pushes the keycode to the keyboard stack.

#### Returns

Returns nothing. If there is an error it returns

Definition at line 77 of file [keyboard.c](#).

Here is the call graph for this function:



### 4.21.2.4 keyboard\_handler\_2()

```
void keyboard_handler_2 ( )
```

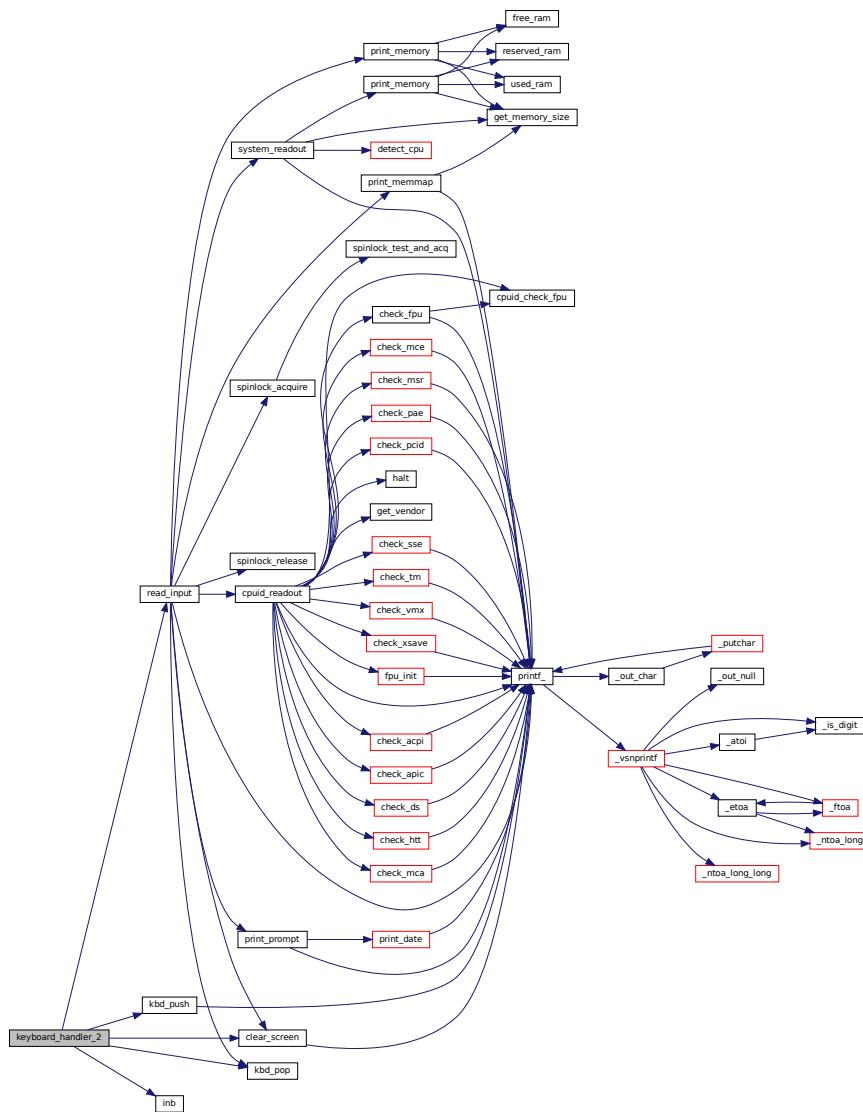
Keyboard handler for 2.0 and later.

#### Returns

none

Definition at line 135 of file [keyboard.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.21.2.5 keyboard\_init()

```
void keyboard_init ( )
```

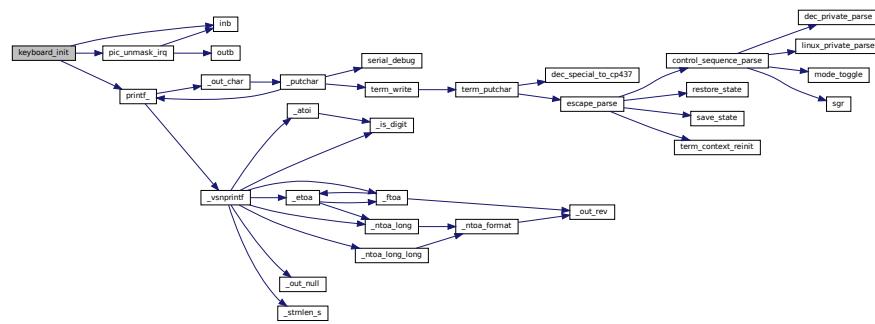
Initialize the keyboard. Unmask IRQ and read keyboard data.

##### Returns

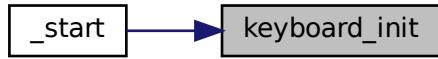
0 on success non - zero

Definition at line 243 of file [keyboard.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



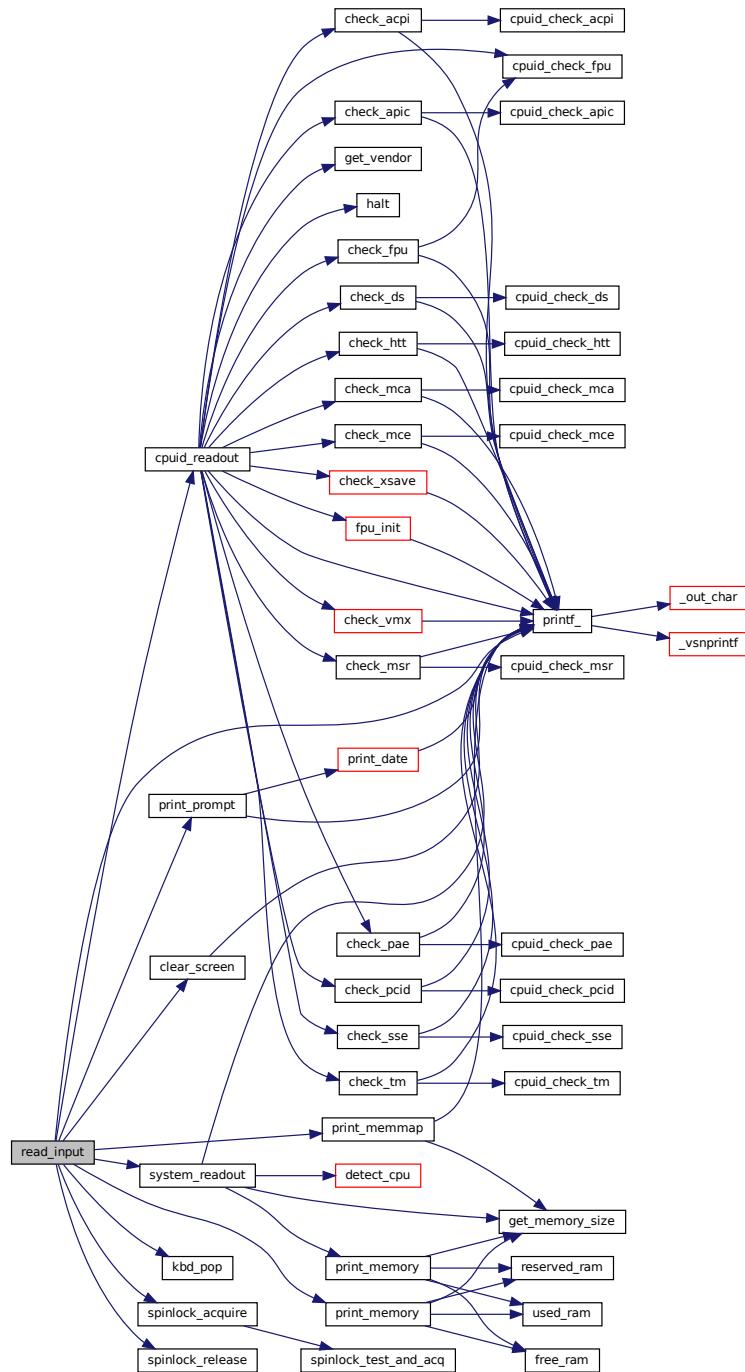
#### 4.21.2.6 read\_input()

```
void read_input ( )
```

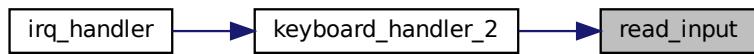
Read input from the keyboard.

Definition at line 173 of file [keyboard.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

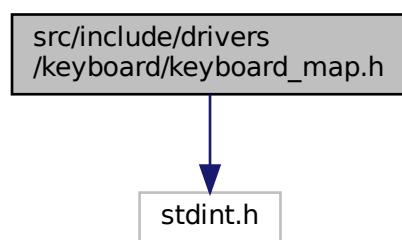


## 4.22 keyboard.h

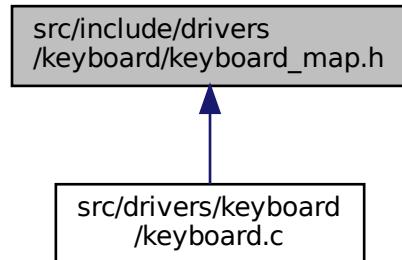
```
00001 #pragma once
00002
00003 #define KEYBOARD_DATA_PORT 0x60
00004 #define KEYBOARD_STATUS_PORT 0x64
00005
00006 void keyboard_handler();
00007
00008 void keyboard_init();
00009
00010 char k_getchar();
00011
00012 char kbd_pop();
00013
00014 void keyboard_handler_2();
00015
00016 void read_input();
```

## 4.23 src/include/drivers/keyboard/keyboard\_map.h File Reference

```
#include <stdint.h>
Include dependency graph for keyboard_map.h:
```



This graph shows which files directly or indirectly include this file:



## Variables

- const char [keyboard\\_map](#) [128]
- char [keyboard\\_charmap](#) [128]

### 4.23.1 Variable Documentation

#### 4.23.1.1 [keyboard\\_charmap](#)

```
char keyboard_charmap[128]
```

Definition at line [156](#) of file [Keyboard\\_map.h](#).

#### 4.23.1.2 [keyboard\\_map](#)

```
const char keyboard_map[128]
```

Definition at line [4](#) of file [Keyboard\\_map.h](#).

## 4.24 keyboard\_map.h

```
00001 #pragma once
00002 #include <stdint.h>
00003
00004 const char keyboard_map[128] = {
00005     // ----- 0 to 9 -----
00006     ',',
00007     '// escape key
00008     '1',
00009     '2',
00010     '3',
00011     '4',
00012     '5',
00013     '6',
00014     '7',
00015     '8',
00016     // ----- 10 to 19 -----
00017     '9',
00018     '0',
00019     '-',
00020     '=',
00021     '// Backspace
00022     '// Tab
00023     'q',
00024     'w',
00025     'e',
00026     'r',
00027     // ----- 20 to 29 -----
00028     't',
00029     'y',
00030     'u',
00031     'i',
00032     'o',
00033     'p',
00034     '[',
00035     ']',
00036     '// Enter
00037     '// left Ctrl
00038     // ----- 30 to 39 -----
00039     'a',
00040     's',
00041     'd',
00042     'f',
00043     'g',
00044     'h',
00045     'j',
00046     'k',
00047     'l',
00048     ';',
00049     // ----- 40 to 49 -----
00050     '/',
00051     '/',
00052     '// left Shift
00053     '/',
00054     'z',
00055     'x',
00056     'c',
00057     'v',
00058     'b',
00059     'n',
00060     // ----- 50 to 59 -----
00061     'm',
00062     '/',
00063     '/',
00064     '// slash, or numpad slash if preceded by keycode 224
00065     '// right Shift
00066     '// numpad asterisk
00067     '// left Alt
00068     '// Spacebar
00069     '/',
00070     '// F1
00071     // ----- 60 to 69 -----
00072     '// F2
00073     '// F3
00074     '// F4
00075     '// F5
00076     '// F6
00077     '// F7
00078     '// F8
00079     '// F9
00080     '// F10
00081     '/',
00082     // ----- 70 to 79 -----
00083     '// scroll lock
00084     '7', // numpad 7, HOME key if preceded by keycode 224
00085     '8', // numpad 8, up arrow if preceded by keycode 224
```

```

00086 '9', // numpad 9, PAGE UP key if preceded by keycode 224
00087 '-', // numpad hyphen
00088 '4', // numpad 4, left arrow if preceded by keycode 224
00089 '5', // numpad 5
00090 '6', // numpad 6, right arrow if preceded by keycode 224
00091 '/',
00092 '1', // numpad 1, END key if preceded by keycode 224
00093 // ----- 80 to 89 -----
00094 '2', // numpad 2, down arrow if preceded by keycode 224
00095 '3', // numpad 3, PAGE DOWN key if preceded by keycode 224
00096 '0', // numpad 0, INSERT key if preceded by keycode 224
00097 '.', // numpad dot, DELETE key if preceded by keycode 224
00098 '/',
00099 '/',
00100 '/',
00101 '/',
00102 '/',
00103 '/',
00104 // ----- 90 to 99 -----
00105 '/',
00106 '/',
00107 '/',
00108 '/',
00109 '/',
00110 '/',
00111 '/',
00112 '/',
00113 '/',
00114 '/',
00115 // ----- 100 to 109 -----
00116 '/',
00117 '/',
00118 '/',
00119 '/',
00120 '/',
00121 '/',
00122 '/',
00123 '/',
00124 '/',
00125 '/',
00126 // ----- 110 to 119 -----
00127 '/',
00128 '/',
00129 '/',
00130 '/',
00131 '/',
00132 '/',
00133 '/',
00134 '/',
00135 '/',
00136 '/',
00137 // ----- 120-127 -----
00138 '/',
00139 '/',
00140 '/',
00141 '/',
00142 '/',
00143 '/',
00144 '/',
00145 '/',
00146 };
00147 // Right control, right alt seem to send
00148 // keycode 224, then the left control/alt keycode
00149 // Arrow keys also send two interrupts, one 224 and then their actual code
00150 // Same for numpad enter
00151 // 197: Num Lock
00152 // 157: Pause|Break (followed by 197?)
00153 // Clicking on screen appears to send keycodes 70, 198
00154 // Is this the MARK command or something like that?
00155
00156 char keyboard_charmap[128] = {
00157 // ----- 0 to 9 -----
00158 '/',
00159 '// escape key
00160 '1',
00161 '2',
00162 '3',
00163 '4',
00164 '5',
00165 '6',
00166 '7',
00167 '8',
00168 // ----- 10 to 19 -----
00169 '9',
00170 '0',
00171 '-',
00172 '=',

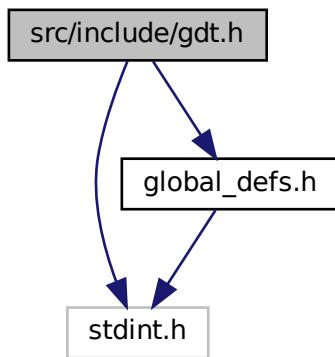
```

```
00173     ' ', // Backspace
00174     ' ', // Tab
00175     'q',
00176     'w',
00177     'e',
00178     'r',
00179     // ----- 20 to 29 -----
00180     't',
00181     'y',
00182     'u',
00183     'i',
00184     'o',
00185     'p',
00186     '[',
00187     ']',
00188     ' ', // Enter
00189     ' ', // left Ctrl
00190     // ----- 30 to 39 -----
00191     'a',
00192     's',
00193     'd',
00194     'f',
00195     'g',
00196     'h',
00197     'j',
00198     'k',
00199     'l',
00200     ';',
00201     // ----- 40 to 49 -----
00202     '/',
00203     '/',
00204     ' ', // left Shift
00205     '/',
00206     'z',
00207     'x',
00208     'c',
00209     'v',
00210     'b',
00211     'n',
00212     // ----- 50 to 59 -----
00213     'm',
00214     '/',
00215     '/',
00216     '/', // slash, or numpad slash if preceded by keycode 224
00217     '/', // right Shift
00218     '*', // numpad asterisk
00219     '/', // left Alt
00220     ' ', // Spacebar
00221     '/',
00222     ' ', // F1
00223     // ----- 60 to 69 -----
00224     ' ', // F2
00225     ' ', // F3
00226     ' ', // F4
00227     ' ', // F5
00228     ' ', // F6
00229     ' ', // F7
00230     ' ', // F8
00231     ' ', // F9
00232     ' ', // F10
00233     '/',
00234     // ----- 70 to 79 -----
00235     ' ', // scroll lock
00236     '7', // numpad 7, HOME key if preceded by keycode 224
00237     '8', // numpad 8, up arrow if preceded by keycode 224
00238     '9', // numpad 9, PAGE UP key if preceded by keycode 224
00239     '–', // numpad hyphen
00240     '4', // numpad 4, left arrow if preceded by keycode 224
00241     '5', // numpad 5
00242     '6', // numpad 6, right arrow if preceded by keycode 224
00243     '/',
00244     '1', // numpad 1, END key if preceded by keycode 224
00245     // ----- 80 to 89 -----
00246     '2', // numpad 2, down arrow if preceded by keycode 224
00247     '3', // numpad 3, PAGE DOWN key if preceded by keycode 224
00248     '0', // numpad 0, INSERT key if preceded by keycode 224
00249     '.', // numpad dot, DELETE key if preceded by keycode 224
00250     '/',
00251     '/',
00252     '/',
00253     '/',
00254     '/',
00255     '/',
00256     // ----- 90 to 99 -----
00257     '/',
00258     '/',
00259     '/'
```

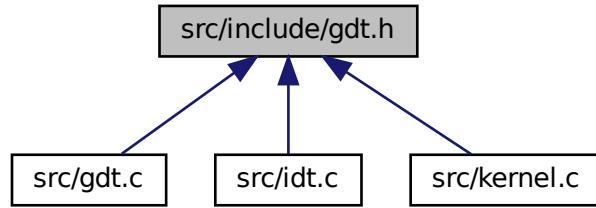
```
00260      ,
00261      ,
00262      ,
00263      ,
00264      ,
00265      ,
00266      ,
00267 // ----- 100 to 109 -----
00268      ,
00269      ,
00270      ,
00271      ,
00272      ,
00273      ,
00274      ,
00275      ,
00276      ,
00277      ,
00278 // ----- 110 to 119 -----
00279      ,
00280      ,
00281      ,
00282      ,
00283      ,
00284      ,
00285      ,
00286      ,
00287      ,
00288      ,
00289 // ----- 120-127 -----
00290      ,
00291      ,
00292      ,
00293      ,
00294      ,
00295      ,
00296      ,
00297      ,
00298 };
```

## 4.25 src/include/gdt.h File Reference

```
#include <stdint.h>
#include "global_defs.h"
Include dependency graph for gdt.h:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `GDT_Desc`
- struct `GDT_Entry`
- struct `GDT_Entry_16`
- struct `GDT_Entry_32`
- struct `TSS_Entry`
- struct `GDT`

## Macros

- #define `GDTAccessDPL(n)` (`n << 5`)
- #define `GDTKernelBaseSelector` 0x28
- #define `GDTUserBaseSelector` 0x48
- #define `GDTTSSSegment` 0x50
- #define `GDTAccess16Code` (`ReadWrite` | `Execute` | `Segments` | `Present`)
- #define `GDTAccess16Data` (`ReadWrite` | `Segments` | `Present`)
- #define `GDTAccess32Code` (`ReadWrite` | `Execute` | `Segments` | `Present`)
- #define `GDTAccess32Data` (`ReadWrite` | `Segments` | `Present`)
- #define `GDTAccessKernelCode` (`ReadWrite` | `Execute` | `Segments` | `Present`)
- #define `GDTAccessKernelData` (`ReadWrite` | `Segments` | `Present`)
- #define `GDTAccessUserCode` (`ReadWrite` | `Execute` | `Segments` | `GDTAccessDPL(3)` | `Present`)
- #define `GDTAccessUserData` (`ReadWrite` | `Segments` | `GDTAccessDPL(3)` | `Present`)

## Enumerations

- enum `GDTAccessFlag` {
   
    `ReadWrite` = (`1 << 1`) , `DC` = (`1 << 2`) , `Execute` = (`1 << 3`) , `Segments` = (`1 << 4`) ,
   
    `Present` = (`1 << 7`) }

## Functions

- void `LoadGDT_Stage1` ()
   
*Stage 1 of GDT loading.*

## Variables

- `uint64_t rsp0`

### 4.25.1 Data Structure Documentation

#### 4.25.1.1 struct GDT\_Desc

Definition at line [30](#) of file `gdt.h`.

##### Data Fields

<code>uint64_t</code>	<code>offset</code>	
<code>uint16_t</code>	<code>size</code>	

#### 4.25.1.2 struct GDT\_Entry

Definition at line [36](#) of file `gdt.h`.

##### Data Fields

<code>uint8_t</code>	<code>access_flag</code>	
<code>uint8_t</code>	<code>base_high</code>	
<code>uint16_t</code>	<code>base_low</code>	
<code>uint8_t</code>	<code>base_middle</code>	
<code>uint8_t</code>	<code>limit_flags</code>	
<code>uint16_t</code>	<code>limit_low</code>	

#### 4.25.1.3 struct GDT\_Entry\_16

Definition at line [46](#) of file `gdt.h`.

##### Data Fields

<code>uint8_t</code>	<code>access_flag</code>	
<code>uint8_t</code>	<code>base_high</code>	
<code>uint16_t</code>	<code>base_low</code>	
<code>uint8_t</code>	<code>base_middle</code>	
<code>uint8_t</code>	<code>limit_flags</code>	
<code>uint16_t</code>	<code>limit_low</code>	

#### 4.25.1.4 struct GDT\_Entry\_32

Definition at line [57](#) of file `gdt.h`.

## Data Fields

uint8_t	access_flag	
uint8_t	base_high	
uint16_t	base_low	
uint8_t	base_middle	
uint8_t	limit_flags	
uint16_t	limit_low	

## 4.25.1.5 struct TSS\_Entry

Definition at line 68 of file [gdt.h](#).

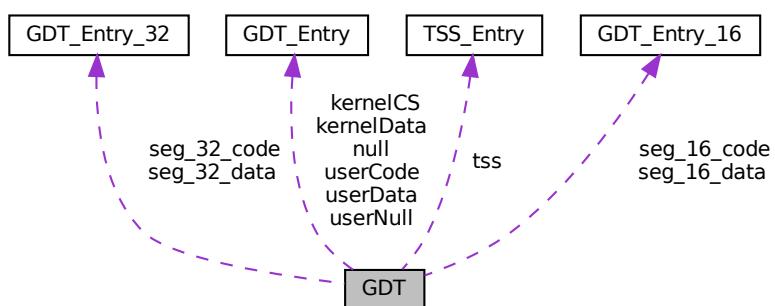
## Data Fields

uint8_t	base_high	
uint16_t	base_low	
uint8_t	base_mid	
uint32_t	base_up	
uint8_t	flags	
uint8_t	flags2	
uint16_t	length	
uint32_t	reserved0	

## 4.25.1.6 struct GDT

Definition at line 80 of file [gdt.h](#).

Collaboration diagram for GDT:



## Data Fields

struct <a href="#">GDT_Entry</a>	kernelCS	

**Data Fields**

struct <a href="#">GDT_Entry</a>	kernelData	
struct <a href="#">GDT_Entry</a>	null	
struct <a href="#">GDT_Entry_16</a>	seg_16_code	
struct <a href="#">GDT_Entry_16</a>	seg_16_data	
struct <a href="#">GDT_Entry_32</a>	seg_32_code	
struct <a href="#">GDT_Entry_32</a>	seg_32_data	
struct <a href="#">TSS_Entry</a>	tss	
struct <a href="#">GDT_Entry</a>	userCode	
struct <a href="#">GDT_Entry</a>	userData	
struct <a href="#">GDT_Entry</a>	userNull	

**4.25.2 Macro Definition Documentation****4.25.2.1 GDTAccess16Code**

```
#define GDTAccess16Code (ReadWrite | Execute | Segments | Present)
```

Definition at line [21](#) of file [gdt.h](#).

**4.25.2.2 GDTAccess16Data**

```
#define GDTAccess16Data (ReadWrite | Segments | Present)
```

Definition at line [22](#) of file [gdt.h](#).

**4.25.2.3 GDTAccess32Code**

```
#define GDTAccess32Code (ReadWrite | Execute | Segments | Present)
```

Definition at line [23](#) of file [gdt.h](#).

**4.25.2.4 GDTAccess32Data**

```
#define GDTAccess32Data (ReadWrite | Segments | Present)
```

Definition at line [24](#) of file [gdt.h](#).

#### 4.25.2.5 GDTAccessDPL

```
#define GDTAccessDPL( n ) (n << 5)
```

Definition at line [6](#) of file [gdt.h](#).

#### 4.25.2.6 GDTAccessKernelCode

```
#define GDTAccessKernelCode (ReadWrite | Execute | Segments | Present)
```

Definition at line [25](#) of file [gdt.h](#).

#### 4.25.2.7 GDTAccessKernelData

```
#define GDTAccessKernelData (ReadWrite | Segments | Present)
```

Definition at line [26](#) of file [gdt.h](#).

#### 4.25.2.8 GDTAccessUserCode

```
#define GDTAccessUserCode (ReadWrite | Execute | Segments | GDTAccessDPL(3) | Present)
```

Definition at line [27](#) of file [gdt.h](#).

#### 4.25.2.9 GDTAccessUserData

```
#define GDTAccessUserData (ReadWrite | Segments | GDTAccessDPL(3) | Present)
```

Definition at line [28](#) of file [gdt.h](#).

#### 4.25.2.10 GDTKernelBaseSelector

```
#define GDTKernelBaseSelector 0x28
```

Definition at line [17](#) of file [gdt.h](#).

#### 4.25.2.11 GDTTSSSegment

```
#define GDTTSSSegment 0x50
```

Definition at line 19 of file [gdt.h](#).

#### 4.25.2.12 GDTUserBaseSelector

```
#define GDTUserBaseSelector 0x48
```

Definition at line 18 of file [gdt.h](#).

### 4.25.3 Enumeration Type Documentation

#### 4.25.3.1 GDTAccessFlag

```
enum GDTAccessFlag
```

Enumerator

ReadWrite	
DC	
Execute	
Segments	
Present	

Definition at line 8 of file [gdt.h](#).

### 4.25.4 Function Documentation

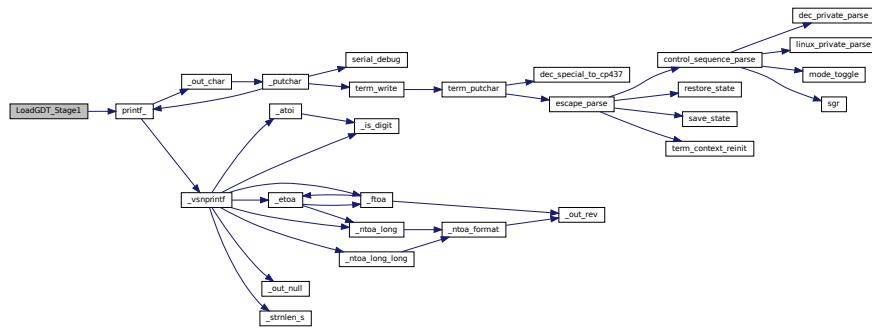
#### 4.25.4.1 LoadGDT\_Stage1()

```
void LoadGDT_Stage1 ( )
```

Stage 1 of [GDT](#) loading.

Definition at line 118 of file [gdt.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.25.5 Variable Documentation

### 4.25.5.1 rsp0

```
uint64_t rsp0 [extern]
```

Definition at line 15 of file [gdt.c](#).

## 4.26 gdt.h

```

00001 #pragma once
00002
00003 #include <stdint.h>
00004 #include "global_defs.h"
00005
00006 #define GDTAccessDPL(n) (n << 5)
00007
00008 enum GDTAccessFlag
00009 {
00010     ReadWrite = (1 << 1),
00011     DC = (1 << 2),
00012     Execute = (1 << 3),
00013     Segments = (1 << 4),
00014     Present = (1 << 7),
00015 };
00016
00017 #define GDTKernelBaseSelector 0x28
00018 #define GDTUserBaseSelector 0x48
  
```

```

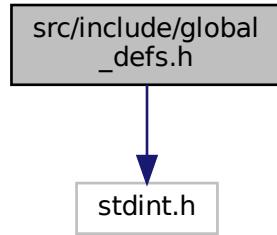
00019 #define GDTSSEGMENT 0x50
00020
00021 #define GDTAccess16Code (ReadWrite | Execute | Segments | Present)
00022 #define GDTAccess16Data (ReadWrite | Segments | Present)
00023 #define GDTAccess32Code (ReadWrite | Execute | Segments | Present)
00024 #define GDTAccess32Data (ReadWrite | Segments | Present)
00025 #define GDTAccessKernelCode (ReadWrite | Execute | Segments | Present)
00026 #define GDTAccessKernelData (ReadWrite | Segments | Present)
00027 #define GDTAccessUserCode (ReadWrite | Execute | Segments | GDTAccessDPL(3) | Present)
00028 #define GDTAccessUserData (ReadWrite | Segments | GDTAccessDPL(3) | Present)
00029
00030 typedef struct PACKED GDT_Desc
00031 {
00032     uint16_t size;
00033     uint64_t offset;
00034 };
00035
00036 typedef struct PACKED GDT_Entry
00037 {
00038     uint16_t limit_low;
00039     uint16_t base_low;
00040     uint8_t base_middle;
00041     uint8_t access_flag;
00042     uint8_t limit_flags;
00043     uint8_t base_high;
00044 };
00045
00046 typedef struct PACKED GDT_Entry_16
00047 {
00048     uint16_t limit_low;
00049     uint16_t base_low;
00050     uint8_t base_middle;
00051     uint8_t access_flag;
00052     uint8_t limit_flags;
00053     uint8_t base_high;
00054 };
00055
00056
00057 typedef struct PACKED GDT_Entry_32
00058 {
00059     uint16_t limit_low;
00060     uint16_t base_low;
00061     uint8_t base_middle;
00062     uint8_t access_flag;
00063     uint8_t limit_flags;
00064     uint8_t base_high;
00065 };
00066
00067
00068 typedef struct PACKED TSS_Entry
00069 {
00070     uint16_t length;
00071     uint16_t base_low;
00072     uint8_t base_mid;
00073     uint8_t flags;
00074     uint8_t flags2;
00075     uint8_t base_high;
00076     uint32_t base_up;
00077     uint32_t reserved0;
00078 };
00079
00080 struct PACKED ALIGN_4K GDT
00081 {
00082     struct GDT_Entry null;
00083     struct GDT_Entry_16 seg_16_code;
00084     struct GDT_Entry_16 seg_16_data;
00085     struct GDT_Entry_32 seg_32_code;
00086     struct GDT_Entry_32 seg_32_data;
00087     struct GDT_Entry kernelCS;
00088     struct GDT_Entry kernelData;
00089     struct GDT_Entry userNull;
00090     struct GDT_Entry userData;
00091     struct GDT_Entry userCode;
00092     struct TSS_Entry tss;
00093 };
00094
00095 extern uint64_t rsp0;
00096
00097 void LoadGDTStage1();

```

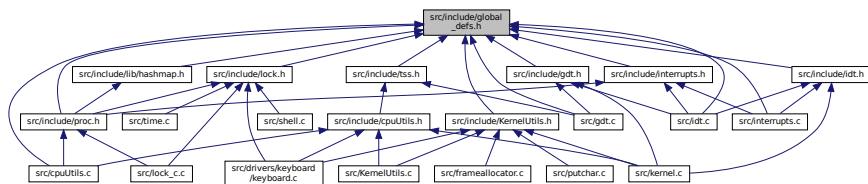
## 4.27 src/include/global\_defs.h File Reference

```
#include <stdint.h>
```

Include dependency graph for global\_defs.h:



This graph shows which files directly or indirectly include this file:



## Macros

- `#define PACKED __attribute__((packed))`
- `#define ALIGN_4K __attribute__((aligned(0x1000)))`
- `#define ALIGN_16BIT __attribute__((aligned(0x10)))`
- `#define CAS __sync_bool_compare_and_swap`

## Typedefs

- `typedef int32_t mode_t`

### 4.27.1 Macro Definition Documentation

#### 4.27.1.1 ALIGN\_16BIT

```
#define ALIGN_16BIT __attribute__((aligned(0x10)))
```

Definition at line 8 of file [global\\_defs.h](#).

#### 4.27.1.2 ALIGN\_4K

```
#define ALIGN_4K __attribute__((aligned(0x1000)))
```

Definition at line 6 of file [global\\_defs.h](#).

#### 4.27.1.3 CAS

```
#define CAS __sync_bool_compare_and_swap
```

Definition at line 10 of file [global\\_defs.h](#).

#### 4.27.1.4 PACKED

```
#define PACKED __attribute__((packed))
```

Definition at line 4 of file [global\\_defs.h](#).

### 4.27.2 Typedef Documentation

#### 4.27.2.1 mode\_t

```
typedef int32_t mode_t
```

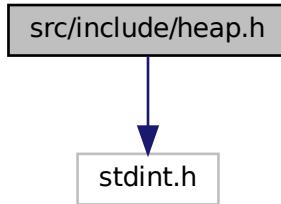
Definition at line 12 of file [global\\_defs.h](#).

## 4.28 global\_defs.h

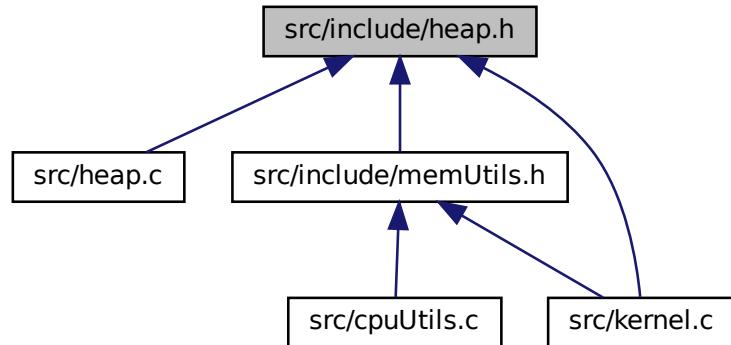
```
00001 #pragma once
00002 #include <stdint.h>
00003
00004 #define PACKED __attribute__((packed))
00005
00006 #define ALIGN_4K __attribute__((aligned(0x1000)))
00007
00008 #define ALIGN_16BIT __attribute__((aligned(0x10)))
00009
00010 #define CAS __sync_bool_compare_and_swap
00011
00012 typedef int32_t mode_t;
```

## 4.29 src/include/heap.h File Reference

```
#include <stdint.h>
Include dependency graph for heap.h:
```



This graph shows which files directly or indirectly include this file:



### Data Structures

- struct [\\_KHEAPBLOCKBM](#)
- struct [\\_KHEAPBM](#)

### Typedefs

- typedef struct [\\_KHEAPBLOCKBM](#) KHEAPBLOCKBM
- typedef struct [\\_KHEAPBM](#) KHEAPBM

## Functions

- void `k_heapBMInit (KHEAPBM *heap)`
- int `k_heapBMAAddBlock (KHEAPBM *heap, uintptr_t addr, uint64_t size, uint64_t bsize)`
- int `k_heapBMAAddBlockEx (KHEAPBM *heap, uintptr_t addr, uint64_t size, uint64_t bsize, KHEAPBLOCKBM *b, uint8_t *bm, uint8_t isBMInside)`
- void \* `k_heapBMAAlloc (KHEAPBM *heap, uint64_t size)`
- void `k_heapBMFree (KHEAPBM *heap, void *ptr)`
- uintptr\_t `k_heapBMGetBMSize (uintptr_t size, uint64_t bsize)`
- void \* `k_heapBMAAllocBound (KHEAPBM *heap, uint64_t size, uint64_t mask)`
- void `k_heapBMSet (KHEAPBM *heap, uintptr_t ptr, uintptr_t size, uint8_t rval)`

### 4.29.1 Data Structure Documentation

#### 4.29.1.1 struct \_KHEAPBLOCKBM

Definition at line 4 of file `heap.h`.

Collaboration diagram for `_KHEAPBLOCKBM`:



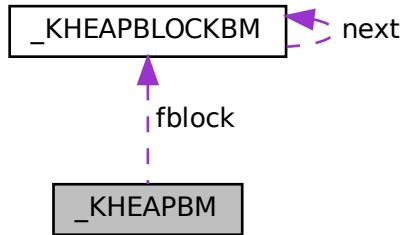
#### Data Fields

<code>uint8_t *</code>	<code>bm</code>	
<code>uint64_t</code>	<code>bsize</code>	
<code>uintptr_t</code>	<code>data</code>	
<code>uint64_t</code>	<code>lfb</code>	
<code>struct _KHEAPBLOCKBM *</code>	<code>next</code>	
<code>uint64_t</code>	<code>size</code>	
<code>uint64_t</code>	<code>used</code>	

#### 4.29.1.2 struct \_KHEAPBM

Definition at line 15 of file `heap.h`.

Collaboration diagram for \_KHEAPBM:



#### Data Fields

<code>KHEAPBLOCKBM *</code>	<code>fblock</code>	<input type="button" value=" "/>
-----------------------------	---------------------	----------------------------------

## 4.29.2 Typedef Documentation

### 4.29.2.1 KHEAPBLOCKBM

```
typedef struct _KHEAPBLOCKBM KHEAPBLOCKBM
```

### 4.29.2.2 KHEAPBM

```
typedef struct _KHEAPBM KHEAPBM
```

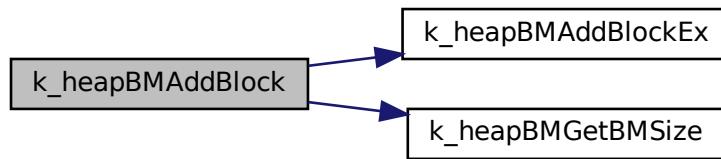
## 4.29.3 Function Documentation

#### 4.29.3.1 k\_heapBMAddBlock()

```
int k_heapBMAddBlock (
    KHEAPBM * heap,
    uintptr_t addr,
    uint64_t size,
    uint64_t bsize )
```

Definition at line 11 of file [heap.c](#).

Here is the call graph for this function:



#### 4.29.3.2 k\_heapBMAddBlockEx()

```
int k_heapBMAddBlockEx (
    KHEAPBM * heap,
    uintptr_t addr,
    uint64_t size,
    uint64_t bsize,
    KHEAPBLOCKBM * b,
    uint8_t * bm,
    uint8_t isBMSInside )
```

Definition at line 29 of file [heap.c](#).

Here is the caller graph for this function:



#### 4.29.3.3 k\_heapBMAalloc()

```
void* k_heapBMAalloc (
    KHEAPBM * heap,
    uint64_t size )
```

Definition at line 77 of file [heap.c](#).

Here is the call graph for this function:



#### 4.29.3.4 k\_heapBMAallocBound()

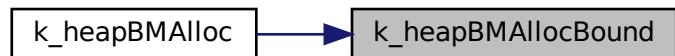
```
void* k_heapBMAallocBound (
    KHEAPBM * heap,
    uint64_t size,
    uint64_t mask )
```

Definition at line 82 of file [heap.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.29.3.5 k\_heapBMFree()

```
void k_heapBMFree (
    KHEAPBM * heap,
    void * ptr )
```

Definition at line 225 of file [heap.c](#).

#### 4.29.3.6 k\_heapBMGetBMSize()

```
uintptr_t k_heapBMGetBMSize (
    uintptr_t size,
    uint64_t bsize )
```

Definition at line 24 of file [heap.c](#).

Here is the caller graph for this function:



#### 4.29.3.7 k\_heapBMInit()

```
void k_heapBMInit (
    KHEAPBM * heap )
```

Definition at line 6 of file [heap.c](#).

#### 4.29.3.8 k\_heapBMSet()

```
void k_heapBMSet (
    KHEAPBM * heap,
    uintptr_t ptr,
    uintptr_t size,
    uint8_t rval )
```

Definition at line 154 of file [heap.c](#).

## 4.30 heap.h

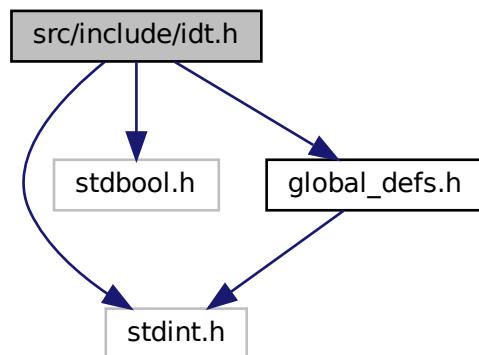
```

00001 #pragma once
00002 #include <stdint.h>
00003
00004 typedef struct _KHEAPBLOCKBM
00005 {
00006     struct _KHEAPBLOCKBM *next;
00007     uint64_t size;
00008     uint64_t used;
00009     uint64_t bsize;
00010     uint64_t lfb;
00011     uintptr_t data;
00012     uint8_t *bm;
00013 } KHEAPBLOCKBM;
00014
00015 typedef struct _KHEAPBM
00016 {
00017     KHEAPBLOCKBM *fblock;
00018 } KHEAPBM;
00019
00020 void k_heapBMAInit(KHEAPBM *heap);
00021 int k_heapBMAddBlock(KHEAPBM *heap, uintptr_t addr, uint64_t size, uint64_t bsize);
00022 int k_heapBMAddBlockEx(KHEAPBM *heap, uintptr_t addr, uint64_t size, uint64_t bsize, KHEAPBLOCKBM *b,
00023     uint8_t *bm, uint8_t isBMInside);
00024 void *k_heapBMAalloc(KHEAPBM *heap, uint64_t size);
00025 void k_heapBMFree(KHEAPBM *heap, void *ptr);
00026 uintptr_t k_heapBMGetBMSize(uintptr_t size, uint64_t bsize);
00027 void *k_heapBMAallocBound(KHEAPBM *heap, uint64_t size, uint64_t mask);
00028 void k_heapBMSet(KHEAPBM *heap, uintptr_t ptr, uintptr_t size, uint8_t rval);

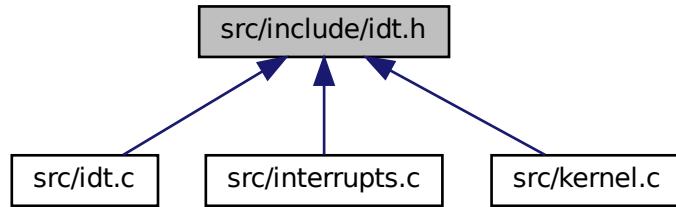
```

## 4.31 src/include/idt.h File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include "global_defs.h"
Include dependency graph for idt.h:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [\\_\\_attribute\\_\\_](#)

## Macros

- #define IDT\_MAX\_DESCRIPTORs 256
- #define IDT\_CPU\_EXCEPTION\_COUNT 32
- #define IDT\_HDW\_INTERRUPT\_COUNT 17
- #define IDT\_DESCRIPTOR\_X16\_INTERRUPT 0x06
- #define IDT\_DESCRIPTOR\_X16\_TRAP 0x07
- #define IDT\_DESCRIPTOR\_X32\_TASK 0x05
- #define IDT\_DESCRIPTOR\_X32\_INTERRUPT 0x0E
- #define IDT\_DESCRIPTOR\_X32\_TRAP 0x0F
- #define IDT\_DESCRIPTOR\_RING1 0x40
- #define IDT\_DESCRIPTOR\_RING2 0x20
- #define IDT\_DESCRIPTOR\_RING3 0x60
- #define IDT\_DESCRIPTOR\_PRESENT 0x80
- #define IDT\_DESCRIPTOR\_EXCEPTION (IDT\_DESCRIPTOR\_X32\_INTERRUPT | IDT\_DESCRIPTOR\_PRESENT)
- #define IDT\_DESCRIPTOR\_EXTERNAL (IDT\_DESCRIPTOR\_X32\_INTERRUPT | IDT\_DESCRIPTOR\_PRESENT)
- #define IDT\_DESCRIPTOR\_CALL (IDT\_DESCRIPTOR\_X32\_INTERRUPT | IDT\_DESCRIPTOR\_PRESENT | IDT\_DESCRIPTOR\_RING3)

## Functions

- void [idt\\_reload](#) (idtr\_t \*idtr)
- uint8\_t [idt\\_allocate\\_vector](#) (void)
- void [idt\\_free\\_vector](#) (uint8\_t vector)
- void [idt\\_set\\_descriptor](#) (uint8\_t vector, uintptr\_t isr, uint8\_t flags, uint8\_t ist)
- void [idt\\_init](#) (void)

### 4.31.1 Data Structure Documentation

#### 4.31.1.1 struct \_\_attribute\_\_

Definition at line 27 of file [idt.h](#).

**Data Fields**

uint8_t	attributes	
uint64_t	base	
uint32_t	base_high	
uint16_t	base_low	
uint16_t	base_mid	
uint16_t	cs	
uint8_t	ist	
uint16_t	limit	
uint32_t	rsv0	

**4.31.2 Macro Definition Documentation****4.31.2.1 IDT\_CPU\_EXCEPTION\_COUNT**

```
#define IDT_CPU_EXCEPTION_COUNT 32
```

Definition at line [10](#) of file [idt.h](#).

**4.31.2.2 IDT\_DESCRIPTOR\_CALL**

```
#define IDT_DESCRIPTOR_CALL (IDT_DESCRIPTOR_X32_INTERRUPT | IDT_DESCRIPTOR_PRESENT | IDT_DESCRIPTOR_RING3)
```

Definition at line [25](#) of file [idt.h](#).

**4.31.2.3 IDT\_DESCRIPTOR\_EXCEPTION**

```
#define IDT_DESCRIPTOR_EXCEPTION (IDT_DESCRIPTOR_X32_INTERRUPT | IDT_DESCRIPTOR_PRESENT)
```

Definition at line [23](#) of file [idt.h](#).

**4.31.2.4 IDT\_DESCRIPTOR\_EXTERNAL**

```
#define IDT_DESCRIPTOR_EXTERNAL (IDT_DESCRIPTOR_X32_INTERRUPT | IDT_DESCRIPTOR_PRESENT)
```

Definition at line [24](#) of file [idt.h](#).

#### 4.31.2.5 IDT\_DESCRIPTOR\_PRESENT

```
#define IDT_DESCRIPTOR_PRESENT 0x80
```

Definition at line 21 of file [idt.h](#).

#### 4.31.2.6 IDT\_DESCRIPTOR\_RING1

```
#define IDT_DESCRIPTOR_RING1 0x40
```

Definition at line 18 of file [idt.h](#).

#### 4.31.2.7 IDT\_DESCRIPTOR\_RING2

```
#define IDT_DESCRIPTOR_RING2 0x20
```

Definition at line 19 of file [idt.h](#).

#### 4.31.2.8 IDT\_DESCRIPTOR\_RING3

```
#define IDT_DESCRIPTOR_RING3 0x60
```

Definition at line 20 of file [idt.h](#).

#### 4.31.2.9 IDT\_DESCRIPTOR\_X16\_INTERRUPT

```
#define IDT_DESCRIPTOR_X16_INTERRUPT 0x06
```

Definition at line 13 of file [idt.h](#).

#### 4.31.2.10 IDT\_DESCRIPTOR\_X16\_TRAP

```
#define IDT_DESCRIPTOR_X16_TRAP 0x07
```

Definition at line 14 of file [idt.h](#).

#### 4.31.2.11 IDT\_DESCRIPTOR\_X32\_INTERRUPT

```
#define IDT_DESCRIPTOR_X32_INTERRUPT 0x0E
```

Definition at line 16 of file [idt.h](#).

#### 4.31.2.12 IDT\_DESCRIPTOR\_X32\_TASK

```
#define IDT_DESCRIPTOR_X32_TASK 0x05
```

Definition at line 15 of file [idt.h](#).

#### 4.31.2.13 IDT\_DESCRIPTOR\_X32\_TRAP

```
#define IDT_DESCRIPTOR_X32_TRAP 0x0F
```

Definition at line 17 of file [idt.h](#).

#### 4.31.2.14 IDT\_HDW\_INTERRUPT\_COUNT

```
#define IDT_HDW_INTERRUPT_COUNT 17
```

Definition at line 11 of file [idt.h](#).

#### 4.31.2.15 IDT\_MAX\_DESCRIPTORS

```
#define IDT_MAX_DESCRIPTORS 256
```

Definition at line 9 of file [idt.h](#).

### 4.31.3 Function Documentation

#### 4.31.3.1 idt\_allocate\_vector()

```
uint8_t idt_allocate_vector (
    void )
```

Definition at line 63 of file [idt.c](#).

### 4.31.3.2 idt\_free\_vector()

```
void idt_free_vector (
    uint8_t vector )
```

Definition at line 77 of file [idt.c](#).

Here is the call graph for this function:

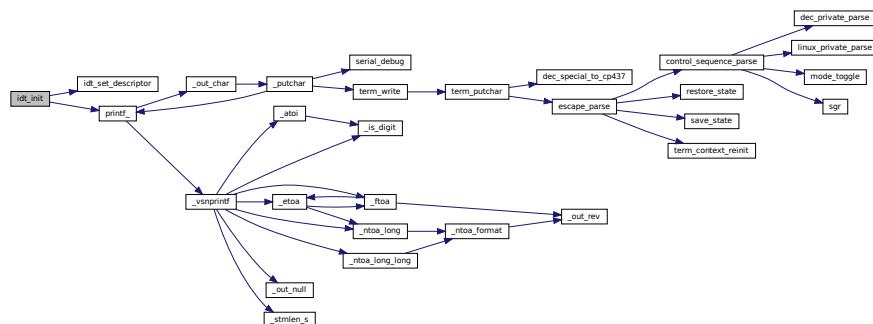


### 4.31.3.3 idt\_init()

```
void idt_init (
    void )
```

Definition at line 33 of file [idt.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 4.31.3.4 idt\_reload()

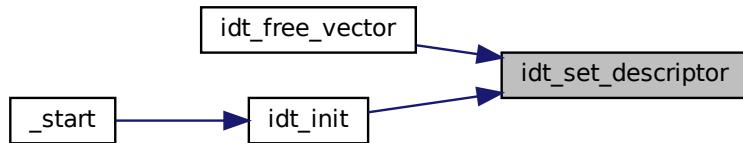
```
void idt_reload (
    idtr_t * idtr )
```

### 4.31.3.5 idt\_set\_descriptor()

```
void idt_set_descriptor (
    uint8_t vector,
    uintptr_t isr,
    uint8_t flags,
    uint8_t ist )
```

Definition at line 20 of file [idt.c](#).

Here is the caller graph for this function:



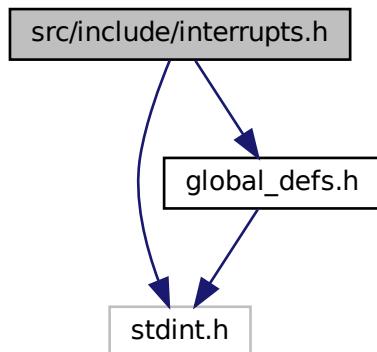
## 4.32 idt.h

```
00001 #pragma once
00002 #include <stdint.h>
00003 #include <stdbool.h>
00004 #include "global_defs.h"
00005
00006 #pragma once
00007 #include <stdint.h>
00008
00009 #define IDT_MAX_DESCRIPTOR 256
00010 #define IDT_CPU_EXCEPTION_COUNT 32
00011 #define IDT_HDW_INTERRUPT_COUNT 17 // including software interrupts called using int
00012
00013 #define IDT_DESCRIPTOR_X16_INTERRUPT 0x06
00014 #define IDT_DESCRIPTOR_X16_TRAP 0x07
00015 #define IDT_DESCRIPTOR_X32_TASK 0x05
00016 #define IDT_DESCRIPTOR_X32_INTERRUPT 0x0E
00017 #define IDT_DESCRIPTOR_X32_TRAP 0x0F
00018 #define IDT_DESCRIPTOR_RING1 0x40
00019 #define IDT_DESCRIPTOR_RING2 0x20
00020 #define IDT_DESCRIPTOR_RING3 0x60
00021 #define IDT_DESCRIPTOR_PRESENT 0x80
00022
00023 #define IDT_DESCRIPTOR_EXCEPTION (IDT_DESCRIPTOR_X32_INTERRUPT | IDT_DESCRIPTOR_PRESENT)
00024 #define IDT_DESCRIPTOR_EXTERNAL (IDT_DESCRIPTOR_X32_INTERRUPT | IDT_DESCRIPTOR_PRESENT)
00025 #define IDT_DESCRIPTOR_CALL (IDT_DESCRIPTOR_X32_INTERRUPT | IDT_DESCRIPTOR_PRESENT |
    IDT_DESCRIPTOR_RING3)
00026
00027 typedef struct
00028 {
00029     uint16_t base_low;
00030     uint16_t cs;
00031     uint8_t ist;
```

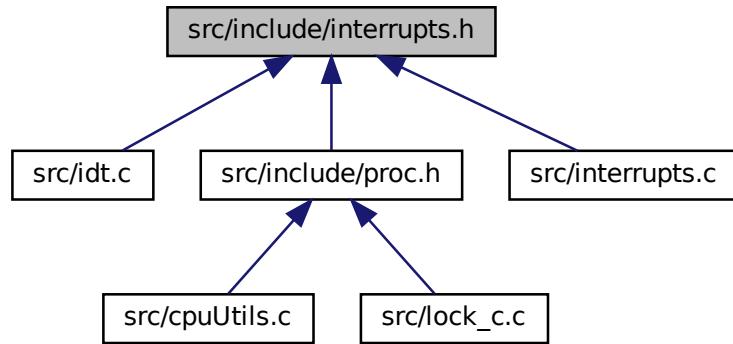
```
00032     uint8_t attributes;
00033     uint16_t base_mid;
00034     uint32_t base_high;
00035     uint32_t rsv0;
00036 } __attribute__((packed)) idt_desc_t;
00037
00038 typedef struct
00039 {
00040     uint16_t limit;
00041     uint64_t base;
00042 } __attribute__((packed)) idtr_t;
00043
00044 void idt_reload(idtr_t *idtr);
00045 uint8_t idt_allocate_vector(void);
00046 void idt_free_vector(uint8_t vector);
00047 void idt_set_descriptor(uint8_t vector, uintptr_t isr, uint8_t flags, uint8_t ist);
00048 void idt_init(void);
```

### 4.33 src/include/interrupts.h File Reference

```
#include <stdint.h>
#include "global_defs.h"
Include dependency graph for interrupts.h:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `isr_xframe_t`
- struct `isr_xframe_t.control_registers`
- struct `isr_xframe_t.general_registers`
- struct `isr_xframe_t.base_frame`

### 4.33.1 Data Structure Documentation

#### 4.33.1.1 struct `isr_xframe_t`

Definition at line 5 of file [interrupts.h](#).

##### Data Fields

<code>struct isr_xframe_t</code>	<code>base_frame</code>	
<code>struct isr_xframe_t</code>	<code>control_registers</code>	
<code>struct isr_xframe_t</code>	<code>general_registers</code>	

#### 4.33.1.2 struct `isr_xframe_t.control_registers`

Definition at line 7 of file [interrupts.h](#).

##### Data Fields

<code>uint64_t</code>	<code>cr0</code>	
<code>uint64_t</code>	<code>cr2</code>	
<code>uint64_t</code>	<code>cr3</code>	
<code>uint64_t</code>	<code>cr4</code>	

### 4.33.1.3 struct isr\_xframe\_t.general\_registers

Definition at line 15 of file [interrupts.h](#).

#### Data Fields

uint64_t	rax	
uint64_t	rbx	
uint64_t	rcx	
uint64_t	rdi	
uint64_t	rdx	
uint64_t	rsi	

### 4.33.1.4 struct isr\_xframe\_t.base\_frame

Definition at line 25 of file [interrupts.h](#).

#### Data Fields

uint64_t	cs	
uint64_t	dss	
uint64_t	error_code	
uint64_t	rbp	
uint64_t	rflags	
uint64_t	rip	
uint64_t	rsp	
uint64_t	vector	

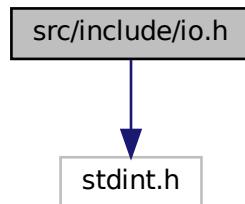
## 4.34 interrupts.h

```
00001 #pragma once
00002 #include <stdint.h>
00003 #include "global_defs.h"
00004
00005 typedef struct
00006 {
00007     struct
00008     {
00009         uint64_t cr4;
00010         uint64_t cr3;
00011         uint64_t cr2;
00012         uint64_t cr0;
00013     } control_registers;
00014
00015     struct
00016     {
00017         uint64_t rdi;
00018         uint64_t rsi;
00019         uint64_t rdx;
00020         uint64_t rcx;
00021         uint64_t rbx;
00022         uint64_t rax;
00023     } general_registers;
00024
00025     struct
00026     {
00027         uint64_t rbp;
00028         uint64_t vector;
00029         uint64_t error_code;
00030         uint64_t rip;
```

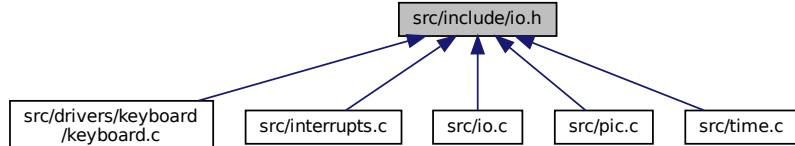
```
00031     uint64_t cs;
00032     uint64_t rflags;
00033     uint64_t rsp;
00034     uint64_t dss;
00035 } base_frame;
00036 } isr_xframe_t;
```

## 4.35 src/include/io.h File Reference

```
#include <stdint.h>
Include dependency graph for io.h:
```



This graph shows which files directly or indirectly include this file:



### Macros

- #define \_IO\_H

### Functions

- uint8\_t **inb** (uint16\_t port)
- void **outb** (uint16\_t port, uint8\_t data)
- void **io\_wait** (void)

#### 4.35.1 Macro Definition Documentation

### 4.35.1.1 \_IO\_H

```
#define _IO_H
```

Definition at line 5 of file [io.h](#).

## 4.35.2 Function Documentation

### 4.35.2.1 inb()

```
uint8_t inb (
    uint16_t port )
```

**inb:** Read a byte from an I/O port.

#### Parameters

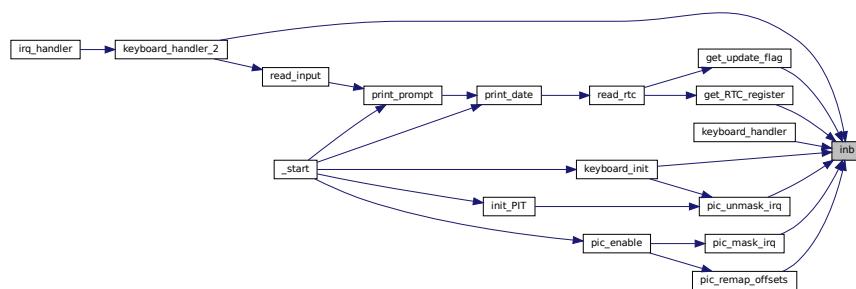
<i>port</i>	The address of the I/O port
-------------	-----------------------------

#### Returns

The read byte

Definition at line 4 of file [io.c](#).

Here is the caller graph for this function:



### 4.35.2.2 io\_wait()

```
void io_wait (
    void )
```

Definition at line 18 of file [io.c](#).

Here is the call graph for this function:



### 4.35.2.3 outb()

```
void outb (
    uint16_t port,
    uint8_t data )
```

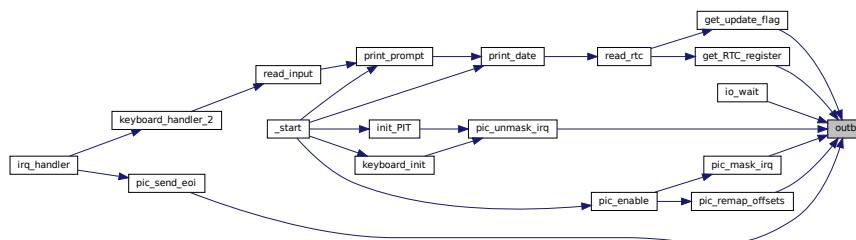
**outb:** Sends the given data to the given I/O port. Defined in [io.s](#)

#### Parameters

<i>port</i>	The I/O port to send the data to
<i>data</i>	The data to send to the I/O port

Definition at line 13 of file [io.c](#).

Here is the caller graph for this function:



## 4.36 io.h

```
00001 #pragma once
00002 #include <stdint.h>
00003
00004 #ifndef _IO_H
00005 #define _IO_H
00006
00013 uint8_t inb(uint16_t port);
```

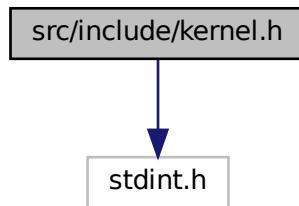
```

00014
00021 void outb(uint16_t port, uint8_t data);
00022
00023 void io_wait(void);
00024
00025 #endif

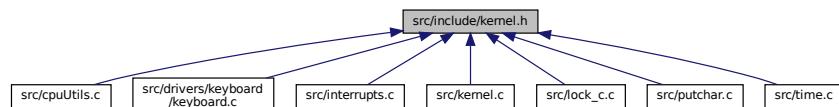
```

## 4.37 src/include/kernel.h File Reference

#include <stdint.h>  
 Include dependency graph for kernel.h:



This graph shows which files directly or indirectly include this file:



## Functions

- void [clear\\_screen \(\)](#)  
*Clears the screen by writing a black line to each of the characters.*
- void [print\\_prompt \(\)](#)  
*Prints the prompt to the user.*

## Variables

- uint32\_t [bootspace](#)
- uint8\_t [kerror\\_mode](#)
- volatile struct [limine\\_terminal\\_request](#) [early\\_term](#)

### 4.37.1 Function Documentation

#### 4.37.1.1 clear\_screen()

```
void clear_screen ( )
```

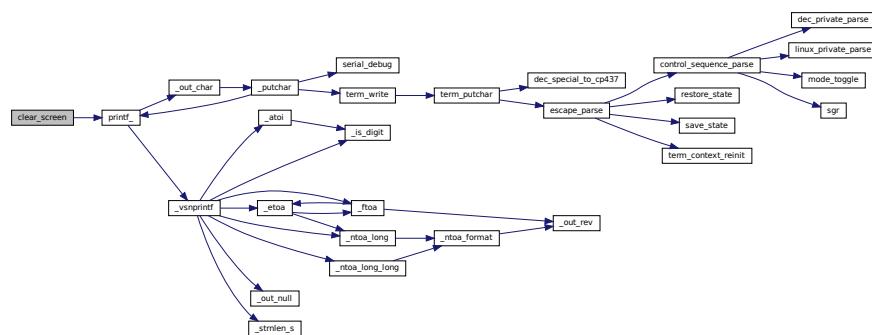
Clears the screen by writing a black line to each of the characters.

##### Returns

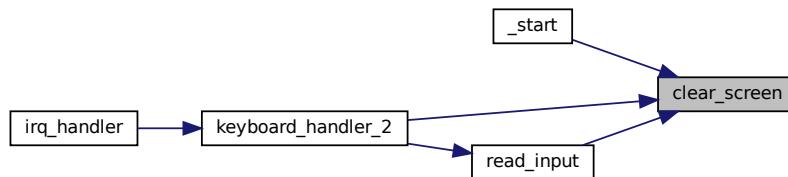
0 on success non - zero

Definition at line 80 of file [kernel.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 4.37.1.2 print\_prompt()

```
void print_prompt( )
```

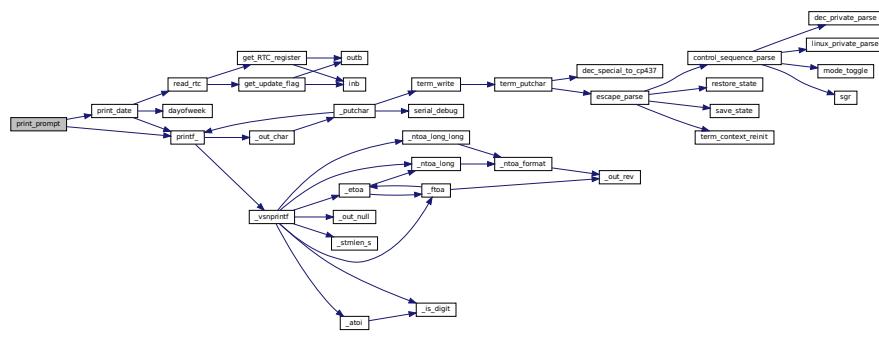
Prints the prompt to the user.

#### Returns

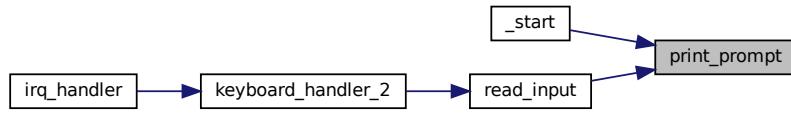
```
void Author : Christian Schafmeister ( 1991 ) Modifications
```

Definition at line 94 of file [kernel.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.37.2 Variable Documentation

### 4.37.2.1 bootspace

```
uint32_t bootspace [extern]
```

Definition at line 68 of file [kernel.c](#).

### 4.37.2.2 early\_term

```
volatile struct limine_terminal_request early_term [extern]
```

Definition at line 39 of file [kernel.c](#).

### 4.37.2.3 kerror\_mode

```
uint8_t kerror_mode [extern]
```

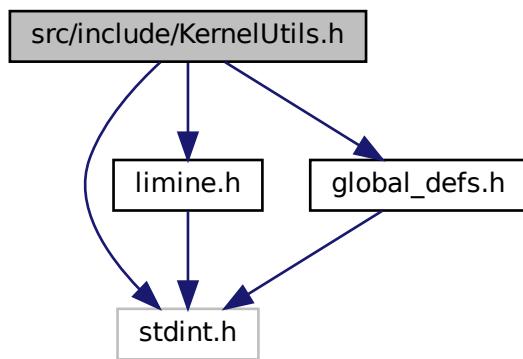
Definition at line 70 of file [kernel.c](#).

## 4.38 kernel.h

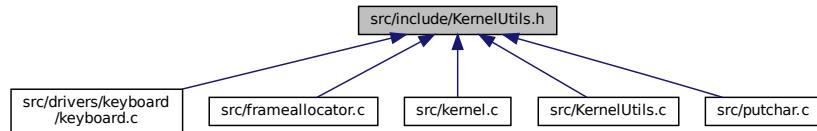
```
00001 #pragma once
00002 #include <stdint.h>
00003
00004 extern uint32_t bootspace;
00005
00006 extern uint8_t kerror_mode;
00007
00008 extern volatile struct limine_terminal_request early_term;
00009
00010 void clear_screen();
00011
00012 void print_prompt();
```

## 4.39 src/include/KernelUtils.h File Reference

```
#include <stdint.h>
#include "limine.h"
#include "global_defs.h"
Include dependency graph for KernelUtils.h:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [kswitches](#)

## Macros

- #define [\\_KERNEL\\_UTILS\\_H](#)

## Functions

- void [print\\_memmap\(\)](#)
- uint64\_t [get\\_memory\\_size\(\)](#)
- void [init\\_memory\(\)](#)
- void [print\\_memory\(\)](#)
- void [system\\_readout\(\)](#)

## Variables

- volatile struct [limine\\_memmap\\_request](#) memmap\_req
- volatile struct [limine\\_framebuffer\\_request](#) fbr\_req
- const struct [kswitches](#) k\_mode
- struct [term\\_context](#) \* term\_context

### 4.39.1 Data Structure Documentation

#### 4.39.1.1 struct kswitches

Definition at line 16 of file [KernelUtils.h](#).

##### Data Fields

<code>uint8_t</code>	<code>acpi_support</code>	
<code>uint8_t</code>	<code>addr_debug</code>	
<code>uint8_t</code>	<code>hw_rng_support</code>	
<code>uint8_t</code>	<code>mem_readout_unit</code>	
<code>uint8_t</code>	<code>sched_debug</code>	
<code>uint8_t</code>	<code>stack_trace_on_fault</code>	
<code>uint8_t</code>	<code>stack_trace_size</code>	

## 4.39.2 Macro Definition Documentation

### 4.39.2.1 \_KERNEL\_UTILS\_H

```
#define _KERNEL_UTILS_H
```

Definition at line 7 of file [KernelUtils.h](#).

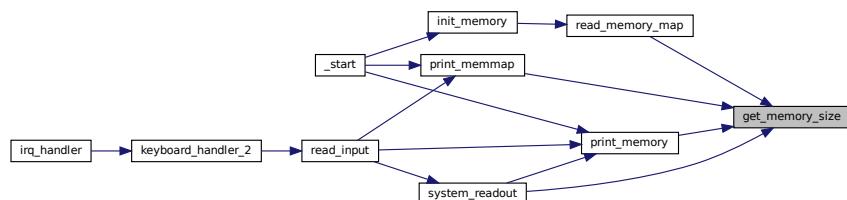
## 4.39.3 Function Documentation

### 4.39.3.1 get\_memory\_size()

```
uint64_t get_memory_size ( )
```

Definition at line 72 of file [KernelUtils.c](#).

Here is the caller graph for this function:

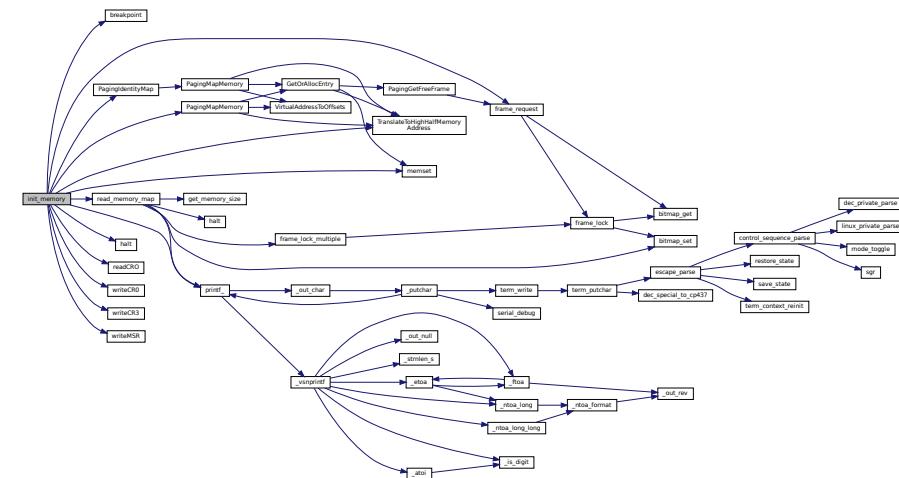


### 4.39.3.2 init\_memory()

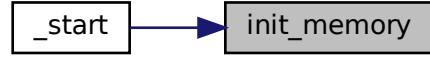
```
void init_memory ( )
```

Definition at line 185 of file [KernelUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

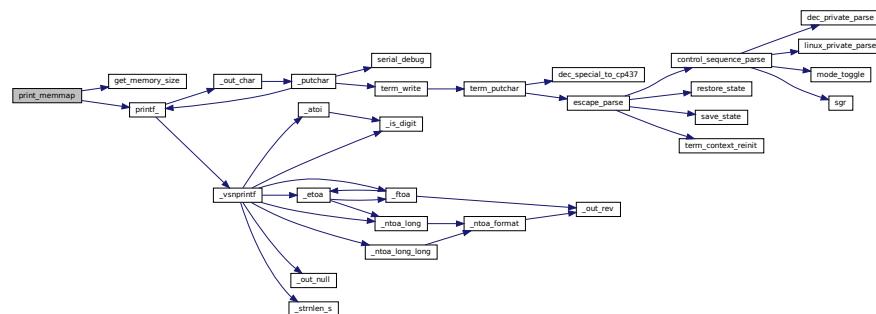


#### 4.39.3.3 print\_memmap()

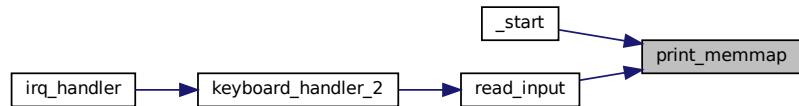
```
void print_memmap ( )
```

Definition at line 89 of file [KernelUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

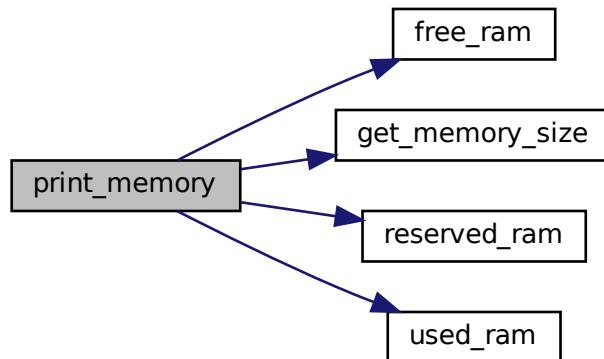


#### 4.39.3.4 print\_memory()

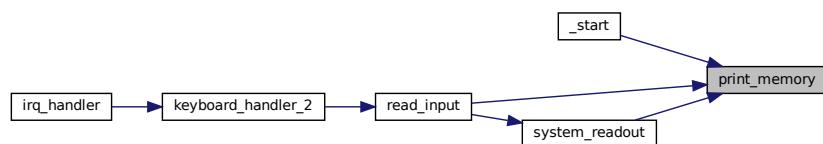
```
void print_memory ( )
```

Definition at line 310 of file [KernelUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

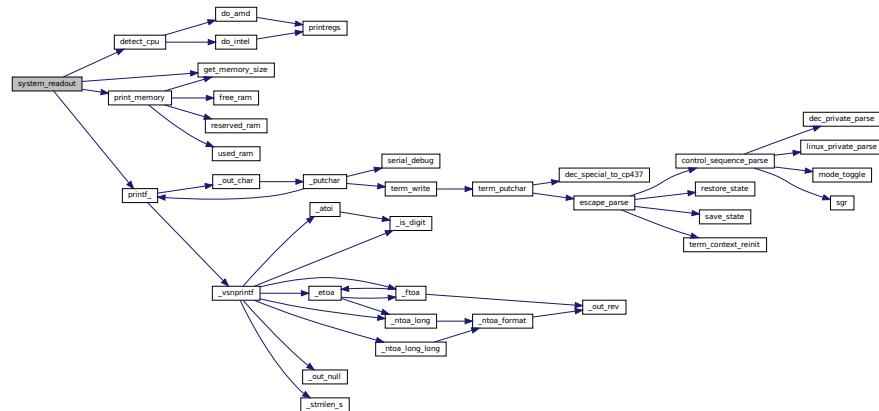


#### 4.39.3.5 system\_readout()

```
void system_readout( )
```

Definition at line 49 of file [KernelUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.39.4 Variable Documentation

##### 4.39.4.1 fbr\_req

```
volatile struct limine_framebuffer_request fbr_req [extern]
```

Definition at line 21 of file [KernelUtils.c](#).

##### 4.39.4.2 k\_mode

```
const struct kswitches k_mode [extern]
```

Definition at line 21 of file [KernelUtils.c](#).

#### 4.39.4.3 memmap\_req

```
volatile struct limine_memmap_request memmap_req [extern]
```

Definition at line 21 of file [KernelUtils.c](#).

#### 4.39.4.4 term\_context

```
struct term_context* term_context [extern]
```

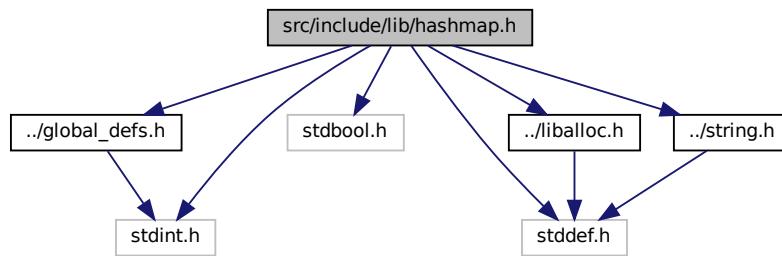
Definition at line 72 of file [kernel.c](#).

## 4.40 KernelUtils.h

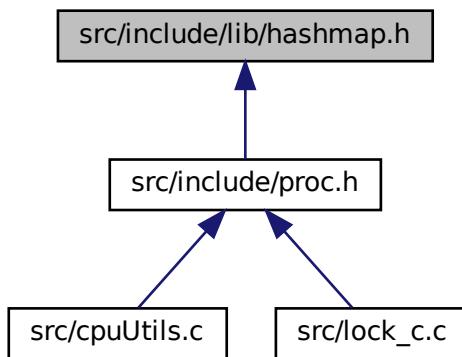
```
00001 #pragma once
00002 #include <stdint.h>
00003 #include "limine.h"
00004 #include "global_defs.h"
00005
00006 #ifndef _KERNEL_UTILS_H
00007 #define _KERNEL_UTILS_H
00008
00009 extern volatile struct limine_memmap_request memmap_req;
00010
00011 extern volatile struct limine_framebuffer_request fbr_req;
00012
00013 extern const struct kswitches k_mode;
00014
00015 // struct contains kernel behavior setting switches. their values are set in KernelUtils.c
00016 struct PACKED kswitches
00017 {
00018     // how many frames back to trace (used as default a differant value can be set in the args)
00019     uint8_t stack_trace_size;
00020
00021     // a value of 1 or 0 or 2, 1 enables 0 disables 2 sets to dump.
00022     uint8_t stack_trace_on_fault;
00023
00024     // a value of 1 or 0, 1 enables 0 disables.
00025     uint8_t acpi_support;
00026
00027     // a value of 1 or 0, 1 enables 0 disables.
00028     uint8_t sched_debug;
00029
00030     // a value of 1 or 0, 1 enables 0 disables. I suggest running with 256mb of ram to avoid a log
00031     // from hell.
00032     uint8_t addr_debug;
00033
00034     uint8_t hw_rng_support;
00035
00036     // 0 gb, 1 mb, 2 kb, 3 b
00037     uint8_t mem_readout_unit;
00038 };
00039
00040 void print_memmap();
00041
00042 uint64_t get_memory_size();
00043
00044 void init_memory();
00045
00046 void print_memory();
00047
00048 void system_readout();
00049
00050 extern struct term_context *term_context;
00051
00052 #endif
```

## 4.41 src/include/lib/hashmap.h File Reference

```
#include <stdint.h>
#include <stddef.h>
#include <stdbool.h>
#include "../global_defs.h"
#include "../liballoc.h"
#include "../string.h"
Include dependency graph for hashmap.h:
```



This graph shows which files directly or indirectly include this file:



### Macros

- `#define _HASHMAP_H`
- `#define HASHMAP_KEY_DATA_MAX 256`
- `#define HASHMAP_INIT(CAP)`
- `#define HASHMAP_DELETE(HASHMAP)`
- `#define HASHMAP_TYPE(TYPE)`
- `#define HASHMAP_GET(HASHMAP, RET, KEY_DATA, KEY_LENGTH)`

- `#define HASHMAP_SGET(HASHMAP, RET, STRING)`
- `#define HASHMAP_REMOVE(HASHMAP, KEY_DATA, KEY_LENGTH)`
- `#define HASHMAP_SREMOVE(HASHMAP, STRING)`
- `#define HASHMAP_INSERT(HASHMAP, KEY_DATA, KEY_LENGTH, ITEM)`
- `#define HASHMAP_SINSERT(HASHMAP, STRING, ITEM)`

## Functions

- `static uint32_t hash (const uint8_t *data, size_t length)`

### 4.41.1 Macro Definition Documentation

#### 4.41.1.1 \_HASHMAP\_H

```
#define _HASHMAP_H
```

Definition at line 3 of file [hashmap.h](#).

#### 4.41.1.2 HASHMAP\_DELETE

```
#define HASHMAP_DELETE (
    HASHMAP )
```

##### Value:

```
do \
{ \
    \
    \
    auto_type HASHMAP_DELETE_hashmap = HASHMAP; \
    \
    \
    if (HASHMAP_DELETE_hashmap->buckets == NULL) \
    { \
        \
        break; \
    } \
    \
    \
    for (size_t HASHMAP_DELETE_i = 0; HASHMAP_DELETE_i < HASHMAP_DELETE_hashmap->cap; \
    HASHMAP_DELETE_i++) \
    { \
        \
        __auto_type HASHMAP_DELETE_bucket = &HASHMAP_DELETE_hashmap->buckets[HASHMAP_DELETE_i]; \
        \
        free(HASHMAP_DELETE_bucket->items); \
    } \
    \
    \
    free(HASHMAP_DELETE_hashmap->buckets); \
    \
} while (0)
```

Definition at line 34 of file [hashmap.h](#).

#### 4.41.1.3 HASHMAP\_GET

```
#define HASHMAP_GET( \
    HASHMAP, \
    RET, \
    KEY_DATA, \
    KEY_LENGTH )
```

Definition at line 71 of file [hashmap.h](#).

#### 4.41.1.4 HASHMAP\_INIT

```
#define HASHMAP_INIT( \
    CAP )
```

**Value:**

```
{ \
    .cap = (CAP), .buckets = NULL \ \
}
```

Definition at line 29 of file [hashmap.h](#).

#### 4.41.1.5 HASHMAP\_INSERT

```
#define HASHMAP_INSERT( \
    HASHMAP, \
    KEY_DATA, \
    KEY_LENGTH, \
    ITEM )
```

Definition at line 167 of file [hashmap.h](#).

#### 4.41.1.6 HASHMAP\_KEY\_DATA\_MAX

```
#define HASHMAP_KEY_DATA_MAX 256
```

Definition at line 27 of file [hashmap.h](#).

#### 4.41.1.7 HASHMAP\_REMOVE

```
#define HASHMAP_REMOVE( \
    HASHMAP, \
    KEY_DATA, \
    KEY_LENGTH )
```

Definition at line 116 of file [hashmap.h](#).

#### 4.41.1.8 HASHMAP\_SGET

```
#define HASHMAP_SGET(
    HASHMAP,
    RET,
    STRING )
```

**Value:**

```
{
    const char *HASHMAP_SGET_string = (STRING);
    HASHMAP_GET(HASHMAP, RET, HASHMAP_SGET_string, strlen(HASHMAP_SGET_string)); \
}
```

Definition at line 111 of file [hashmap.h](#).

#### 4.41.1.9 HASHMAP\_SINSERT

```
#define HASHMAP_SINSERT(
    HASHMAP,
    STRING,
    ITEM )
```

**Value:**

```
do \
{ \
    const char *HASHMAP_SINSERT_string = (STRING);
    HASHMAP_INSERT(HASHMAP, HASHMAP_SINSERT_string, strlen(HASHMAP_SINSERT_string), ITEM); \
} while (0)
```

Definition at line 209 of file [hashmap.h](#).

#### 4.41.1.10 HASHMAP\_SREMOVE

```
#define HASHMAP_SREMOVE(
    HASHMAP,
    STRING )
```

**Value:**

```
{
    const char *HASHMAP_SREMOVE_string = (STRING);
    HASHMAP_REMOVE(HASHMAP, HASHMAP_SREMOVE_string, strlen(HASHMAP_SREMOVE_string)); \
}
```

Definition at line 162 of file [hashmap.h](#).

#### 4.41.1.11 HASHMAP\_TYPE

```
#define HASHMAP_TYPE(
    TYPE )
```

**Value:**

```
struct
{
    size_t cap;
    struct
    {
        size_t cap;
        size_t filled;
        struct
        {
            uint8_t key_data[HASHMAP_KEY_DATA_MAX];
            size_t key_length;
            TYPE item;
        } * items;
    } * buckets;
}
```

Definition at line 54 of file [hashmap.h](#).

### 4.41.2 Function Documentation

#### 4.41.2.1 hash()

```
static uint32_t hash (
    const uint8_t * data,
    size_t length ) [inline], [static]
```

Definition at line 13 of file [hashmap.h](#).

## 4.42 hashmap.h

```
00001 #pragma once
00002 #ifndef _HASHMAP_H
00003 #define _HASHMAP_H
00004
00005 #include <stdint.h>
00006 #include <stddef.h>
00007 #include <stdbool.h>
00008 #include "../global_defs.h"
00009 #include "../liballoc.h"
00010 #include "../string.h"
00011
00012 // sdbm from: http://www.cse.yorku.ca/~oz/hash.html
00013 static inline uint32_t hash(const uint8_t *data, size_t length)
00014 {
00015     const uint8_t *data_u8 = data; // ignore warning lexer is being a cu*
00016     uint32_t hash = 0;
00017
00018     for (size_t i = 0; i < length; i++)
00019     {
00020         uint32_t c = data_u8[i];
00021         hash = c + (hash << 6) + (hash << 16) - hash;
00022     }
00023
00024     return hash;
00025 }
00026
00027 #define HASHMAP_KEY_DATA_MAX 256
00028
00029 #define HASHMAP_INIT(CAP) \
```

```

00030     {
00031         .cap = (CAP), .buckets = NULL \
00032     }
00033
00034 #define HASHMAP_DELETE (HASHMAP)
00035     \
00036     do \
00037     { \
00038         \
00039         if (HASHMAP_DELETE_hashmap->buckets == NULL) \
00040     {
00041         \
00042         break;
00043     }
00044     \
00045     for (size_t HASHMAP_DELETE_i = 0; HASHMAP_DELETE_i < HASHMAP_DELETE_hashmap->cap;
00046 HASHMAP_DELETE_i++) \
00047     {
00048         \
00049         __auto_type HASHMAP_DELETE_bucket = &HASHMAP_DELETE_hashmap->buckets[HASHMAP_DELETE_i];
00050     }
00051     \
00052     free(HASHMAP_DELETE_hashmap->buckets);
00053 } while (0)
00054 #define HASHMAP_TYPE (TYPE)
00055     struct \
00056     {
00057         size_t cap;
00058         struct \
00059         {
00060             size_t cap;
00061             size_t filled;
00062             struct \
00063             {
00064                 uint8_t key_data[HASHMAP_KEY_DATA_MAX];
00065                 size_t key_length;
00066                 TYPE item;
00067             } * items;
00068         } * buckets;
00069     }
00070
00071 #define HASHMAP_GET (HASHMAP, RET, KEY_DATA, KEY_LENGTH) ({ \
00072     __label__ out; \
00073     bool HASHMAP_GET_ok = false; \
00074     \
00075     __auto_type HASHMAP_GET_key_data = KEY_DATA; \
00076     __auto_type HASHMAP_GET_key_length = KEY_LENGTH; \
00077     \
00078     __auto_type HASHMAP_GET_hashmap = HASHMAP; \
00079     \
00080     if (HASHMAP_GET_hashmap->buckets == NULL) \
00081     { \
00082         goto out; \
00083     } \
00084     \
00085     size_t HASHMAP_GET_hash = hash(HASHMAP_GET_key_data, HASHMAP_GET_key_length); \
00086     size_t HASHMAP_GET_index = HASHMAP_GET_hash % HASHMAP_GET_hashmap->cap; \
00087     \
00088     __auto_type HASHMAP_GET_bucket = &HASHMAP_GET_hashmap->buckets[HASHMAP_GET_index]; \
00089     \
00090     for (size_t HASHMAP_GET_i = 0; HASHMAP_GET_i < HASHMAP_GET_bucket->filled; HASHMAP_GET_i++) \
00091     { \
00092         \
00093         if (HASHMAP_GET_key_length != HASHMAP_GET_bucket->items[HASHMAP_GET_i].key_length) \
00094         { \
00095             continue; \
00096         } \
00097         if (memcmp(HASHMAP_GET_key_data,
00098                 HASHMAP_GET_bucket->items[HASHMAP_GET_i].key_data,

```

```

00099             HASHMAP_GET_key_length) == 0)
00100         {
00101             RET = HASHMAP_GET_bucket->items[HASHMAP_GET_i].item;
00102             HASHMAP_GET_ok = true;
00103             break;
00104         }
00105     }
00106 
00107 out:
00108     HASHMAP_GET_ok;
00109 })
00110 
00111 #define HASHMAP_SGET(HASHMAP, RET, STRING) ({ \
00112     const char *HASHMAP_SGET_string = (STRING); \
00113     HASHMAP_GET(HASHMAP, RET, HASHMAP_SGET_string, strlen(HASHMAP_SGET_string)); \
00114 })
00115 
00116 #define HASHMAP_REMOVE(HASHMAP, KEY_DATA, KEY_LENGTH) ({
00117     \
00118     _label__ out; \
00119     \
00120     bool HASHMAP_REMOVE_ok = false; \
00121     \
00122     _auto_type HASHMAP_REMOVE_key_data = KEY_DATA; \
00123     \
00124     _auto_type HASHMAP_REMOVE_key_length = KEY_LENGTH; \
00125     \
00126     if (HASHMAP_REMOVE_hashmap->buckets == NULL) \
00127     { \
00128         goto out; \
00129     } \
00130     \
00131     size_t HASHMAP_REMOVE_hash = hash(HASHMAP_REMOVE_key_data, HASHMAP_REMOVE_key_length); \
00132     size_t HASHMAP_REMOVE_index = HASHMAP_REMOVE_hash % HASHMAP_REMOVE_hashmap->cap; \
00133     \
00134     _auto_type HASHMAP_REMOVE_bucket = &HASHMAP_REMOVE_hashmap->buckets[HASHMAP_REMOVE_index]; \
00135     \
00136     for (size_t HASHMAP_REMOVE_i = 0; HASHMAP_REMOVE_i < HASHMAP_REMOVE_bucket->filled; \
00137     HASHMAP_REMOVE_i++) \
00138     { \
00139         if (HASHMAP_REMOVE_key_length != HASHMAP_REMOVE_bucket->items[HASHMAP_REMOVE_i].key_length) \
00140         { \
00141             continue; \
00142         } \
00143         if (memcmp(HASHMAP_REMOVE_key_data, \
00144             HASHMAP_REMOVE_bucket->items[HASHMAP_REMOVE_i].key_data, \
00145             HASHMAP_REMOVE_key_length) == 0) \
00146         { \
00147             if (HASHMAP_REMOVE_i != HASHMAP_REMOVE_bucket->filled - 1) \
00148             { \
00149                 memcpy(&HASHMAP_REMOVE_bucket->items[HASHMAP_REMOVE_i], \
00150                     &HASHMAP_REMOVE_bucket->items[HASHMAP_REMOVE_bucket->filled - 1], \
00151                     sizeof(*HASHMAP_REMOVE_bucket->items)); \
00152             } \
00153         } \
00154     } \
00155 }

```

```
00151         \
00152         \
00153         \
00154         \
00155     }
00156 }
00157 \
00158 out:
00159     \
00160 })
00161
00162 #define HASHMAP_SREMOVE (HASHMAP, STRING) ({ \
00163     const char *HASHMAP_SREMOVE_string = (STRING); \
00164     HASHMAP_REMOVE (HASHMAP, HASHMAP_SREMOVE_string, strlen (HASHMAP_SREMOVE_string)); \
00165 })
00166
00167 #define HASHMAP_INSERT (HASHMAP, KEY_DATA, KEY_LENGTH, ITEM) \
00168     do \
00169     { \
00170         __auto_type HASHMAP_INSERT_key_data = KEY_DATA; \
00171         __auto_type HASHMAP_INSERT_key_length = KEY_LENGTH; \
00172         \
00173         __auto_type HASHMAP_INSERT_hashmap = HASHMAP; \
00174         if (HASHMAP_INSERT_hashmap->buckets == NULL) \
00175         { \
00176             HASHMAP_INSERT_hashmap->buckets = \
00177                 malloc (HASHMAP_INSERT_hashmap->cap * sizeof (*HASHMAP_INSERT_hashmap->buckets)); \
00178         } \
00179         \
00180         size_t HASHMAP_INSERT_hash = hash (HASHMAP_INSERT_key_data, HASHMAP_INSERT_key_length); \
00181         size_t HASHMAP_INSERT_index = HASHMAP_INSERT_hash % HASHMAP_INSERT_hashmap->cap; \
00182         \
00183         __auto_type HASHMAP_INSERT_bucket = &HASHMAP_INSERT_hashmap->buckets [HASHMAP_INSERT_index]; \
00184         \
00185         if (HASHMAP_INSERT_bucket->cap == 0) \
00186         { \
00187             HASHMAP_INSERT_bucket->cap = 16; \
00188             HASHMAP_INSERT_bucket->items = \
00189                 malloc (HASHMAP_INSERT_bucket->cap * sizeof (*HASHMAP_INSERT_bucket->items)); \
00190         } \
00191         \
00192         if (HASHMAP_INSERT_bucket->filled == HASHMAP_INSERT_bucket->cap) \
00193         { \
00194             HASHMAP_INSERT_bucket->cap *= 2; \
00195             HASHMAP_INSERT_bucket->items = \
00196                 realloc (HASHMAP_INSERT_bucket->items, \
00197                         HASHMAP_INSERT_bucket->cap * sizeof (*HASHMAP_INSERT_bucket->items)); \
00198         }
```

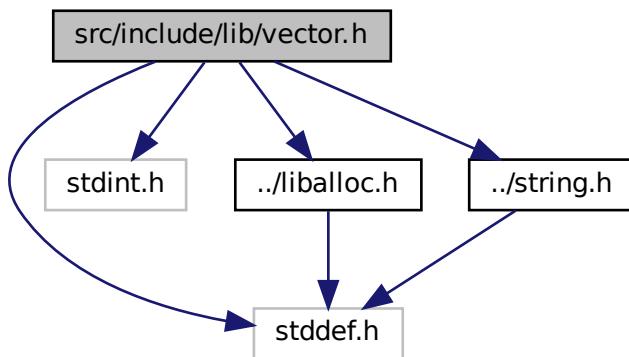
```

00198         }
00199     \
00200     __auto_type HASHMAP_INSERT_item =
00201     &HASHMAP_INSERT_bucket->items[HASHMAP_INSERT_bucket->filled]; \
00202     \
00203     memcpy(HASHMAP_INSERT_item->key_data, HASHMAP_INSERT_key_data, HASHMAP_INSERT_key_length);
00204     \
00205     HASHMAP_INSERT_item->key_length = HASHMAP_INSERT_key_length;
00206     \
00207     HASHMAP_INSERT_item->item = ITEM;
00208     \
00209 } while (0)
00210
00211 #define HASHMAP_SINSERT (HASHMAP, STRING, ITEM)
00212 do
00213 {
00214     const char *HASHMAP_SINSERT_string = (STRING);
00215     HASHMAP_INSERT(HASHMAP, HASHMAP_SINSERT_string, strlen(HASHMAP_SINSERT_string), ITEM);
00216 } while (0)
00217
00218 #endif

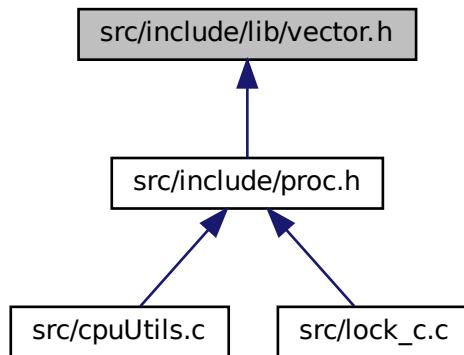
```

## 4.43 src/include/lib/vector.h File Reference

```
#include <stddef.h>
#include <stdint.h>
#include "../liballoc.h"
#include "../string.h"
Include dependency graph for vector.h:
```



This graph shows which files directly or indirectly include this file:



## Macros

- #define VECTOR\_INVALID\_INDEX (-1)
- #define VECTOR\_INIT
- #define VECTOR\_TYPE(TYPE)
- #define VECTOR\_ENSURE\_LENGTH(VEC, LENGTH)
- #define VECTOR\_PUSH\_BACK(VEC, VALUE)
- #define VECTOR\_PUSH\_FRONT(VEC, VALUE) VECTOR\_INSERT(VEC, 0, VALUE)
- #define VECTOR\_INSERT(VEC, IDX, VALUE)
- #define VECTOR\_REMOVE(VEC, IDX)
- #define VECTOR\_ITEM(VEC, IDX)
- #define VECTOR\_FIND(VEC, VALUE)
- #define VECTOR\_REMOVE\_BY\_VALUE(VEC, VALUE)
- #define VECTOR\_FOR\_EACH(VEC, BINDING, ...)

### 4.43.1 Macro Definition Documentation

#### 4.43.1.1 VECTOR\_ENSURE\_LENGTH

```
#define VECTOR_ENSURE_LENGTH(  
    VEC,  
    LENGTH )
```

##### Value:

```
do  
    \  
{  
    \\\_auto_type VECTOR_ENSURE_LENGTH_vec = VEC;  
    \\
```

```

if ((LENGTH) >= VECTOR_ENSURE_LENGTH_vec->capacity)
{
    \
    \
    if (VECTOR_ENSURE_LENGTH_vec->capacity == 0)
    {
        \
        \
        VECTOR_ENSURE_LENGTH_vec->capacity = 8;
    }
    \
    else
    {
        \
        \
        VECTOR_ENSURE_LENGTH_vec->capacity *= 2;
    }
    \
    VECTOR_ENSURE_LENGTH_vec->data = realloc(VECTOR_ENSURE_LENGTH_vec->data,
    \
                                                VECTOR_ENSURE_LENGTH_vec->capacity *
sizeof(*VECTOR_ENSURE_LENGTH_vec->data)); /* NOLINT */
}
}
} while (0)

```

Definition at line 24 of file [vector.h](#).

#### 4.43.1.2 VECTOR\_FIND

```
#define VECTOR_FIND(
    VEC,
    VALUE )
```

**Value:**

```

({ \
    __auto_type VECTOR_FIND_vec = VEC; \
    ssize_t VECTOR_FIND_result = VECTOR_INVALID_INDEX; \
    for (size_t VECTOR_FIND_i = 0; VECTOR_FIND_i < VECTOR_FIND_vec->length; VECTOR_FIND_i++) \
    { \
        if (VECTOR_FIND_vec->data[VECTOR_FIND_i] == (VALUE)) \
        { \
            VECTOR_FIND_result = VECTOR_FIND_i; \
            break; \
        } \
    } \
    VECTOR_FIND_result; \
}) \

```

Definition at line 88 of file [vector.h](#).

#### 4.43.1.3 VECTOR\_FOR\_EACH

```
#define VECTOR_FOR_EACH(
    VEC,
    BINDING,
    ... )
```

**Value:**

```

do \
{ \
    \
}
```

```

__auto_type VECTOR_FOR_EACH_vec = VEC;
    \
    for (size_t VECTOR_FOR_EACH_i = 0; VECTOR_FOR_EACH_i < VECTOR_FOR_EACH_vec->length;
VECTOR_FOR_EACH_i++) \
{
    \
        \
        __auto_type BINDING = &VECTOR_FOR_EACH_vec->data[VECTOR_FOR_EACH_i];
        \
        \
        \
}
} while (0)

```

Definition at line 111 of file [vector.h](#).

#### 4.43.1.4 VECTOR\_INIT

```
#define VECTOR_INIT
```

**Value:**

```

{
    0
}
```

Definition at line 11 of file [vector.h](#).

#### 4.43.1.5 VECTOR\_INSERT

```
#define VECTOR_INSERT(
    VEC,
    IDX,
    VALUE )
```

**Value:**

```

do
{
    \
    \
    __auto_type VECTOR_INSERT_vec = VEC;
    size_t VECTOR_INSERT_index = IDX;
    \
    VECTOR_ENSURE_LENGTH(VEC, VECTOR_INSERT_vec->length);
    \
    for (size_t VECTOR_INSERT_i = VECTOR_INSERT_vec->length; VECTOR_INSERT_i > VECTOR_INSERT_index;
VECTOR_INSERT_i--) \
{
    \
        \
        VECTOR_INSERT_vec->data[VECTOR_INSERT_i] = VECTOR_INSERT_vec->data[VECTOR_INSERT_i - 1];
    \
}
    \
    \
    VECTOR_INSERT_vec->length++;
    \
    VECTOR_INSERT_vec->data[VECTOR_INSERT_index] = VALUE;
}
} while (0)

```

Definition at line 52 of file [vector.h](#).

#### 4.43.1.6 VECTOR\_INVALID\_INDEX

```
#define VECTOR_INVALID_INDEX (-1)
```

Definition at line 9 of file [vector.h](#).

#### 4.43.1.7 VECTOR\_ITEM

```
#define VECTOR_ITEM(
    VEC,
    IDX )
```

**Value:**

```
({
    size_t VECTOR_ITEM_idx = IDX;
    __auto_type VECTOR_ITEM_vec = VEC;
    __auto_type VECTOR_ITEM_result = (typeof(*VECTOR_ITEM_vec->data))VECTOR_INVALID_INDEX; \
    if (VECTOR_ITEM_idx < VECTOR_ITEM_vec->length)
    {
        VECTOR_ITEM_result = VECTOR_ITEM_vec->data[VECTOR_ITEM_idx];
    }
    VECTOR_ITEM_result;
})
```

Definition at line 77 of file [vector.h](#).

#### 4.43.1.8 VECTOR\_PUSH\_BACK

```
#define VECTOR_PUSH_BACK(
    VEC,
    VALUE )
```

**Value:**

```
({
    __auto_type VECTOR_PUSH_BACK_vec = VEC; \
    VECTOR_ENSURE_LENGTH(VEC, VECTOR_PUSH_BACK_vec->length); \
    VECTOR_PUSH_BACK_vec->data[VECTOR_PUSH_BACK_vec->length++] = VALUE; \
    VECTOR_PUSH_BACK_vec->length - 1;
})
```

Definition at line 43 of file [vector.h](#).

#### 4.43.1.9 VECTOR\_PUSH\_FRONT

```
#define VECTOR_PUSH_FRONT(
    VEC,
    VALUE ) VECTOR_INSERT(VEC, 0, VALUE)
```

Definition at line 50 of file [vector.h](#).

#### 4.43.1.10 VECTOR\_REMOVE

```
#define VECTOR_REMOVE(
    VEC,
    IDX )
```

**Value:**

```
do \
{ \
    __auto_type VECTOR_REMOVE_vec = VEC;
    for (size_t VECTOR_REMOVE_i = (IDX); VECTOR_REMOVE_i < VECTOR_REMOVE_vec->length - 1;
        VECTOR_REMOVE_i++) \
    {
        \
        VECTOR_REMOVE_vec->data[VECTOR_REMOVE_i] = VECTOR_REMOVE_vec->data[VECTOR_REMOVE_i + 1];
    }
    \
    VECTOR_REMOVE_vec->length--;
} while (0)
```

Definition at line 66 of file [vector.h](#).

#### 4.43.1.11 VECTOR\_REMOVE\_BY\_VALUE

```
#define VECTOR_REMOVE_BY_VALUE(
    VEC,
    VALUE )
```

**Value:**

```
do \
{ \
    __auto_type VECTOR_REMOVE_BY_VALUE_vec = VEC;
    __auto_type VECTOR_REMOVE_BY_VALUE_v = VALUE;
    size_t VECTOR_REMOVE_BY_VALUE_i = VECTOR_FIND(VECTOR_REMOVE_BY_VALUE_vec, VECTOR_REMOVE_BY_VALUE_v);
    \
    VECTOR_REMOVE(VECTOR_REMOVE_BY_VALUE_vec, VECTOR_REMOVE_BY_VALUE_i);
} while (0)
```

Definition at line 102 of file [vector.h](#).

#### 4.43.1.12 VECTOR\_TYPE

```
#define VECTOR_TYPE(
    TYPE )
```

**Value:**

```
struct \
{ \
    TYPE *data; \
    size_t length; \
    size_t capacity; \
}
```

Definition at line 16 of file [vector.h](#).

## 4.44 vector.h

```

00001 #ifndef _VECTOR_H
00002 #define _VECTOR_H
00003
00004 #include <stddef.h>
00005 #include <stdint.h>
00006 #include "../liballoc.h"
00007 #include "../string.h"
00008
00009 #define VECTOR_INVALID_INDEX (-1)
00010
00011 #define VECTOR_INIT \
00012     {           \
00013         0           \
00014     }
00015
00016 #define VECTOR_TYPE(TYPE) \
00017     struct \
00018     {           \
00019         TYPE *data;           \
00020         size_t length;           \
00021         size_t capacity;           \
00022     }
00023
00024 #define VECTOR_ENSURE_LENGTH(VEC, LENGTH) \
00025     do \
00026     { \
00027         __auto_type VECTOR_ENSURE_LENGTH_vec = VEC; \
00028         if ((LENGTH) >= VECTOR_ENSURE_LENGTH_vec->capacity) \
00029         { \
00030             if (VECTOR_ENSURE_LENGTH_vec->capacity == 0) \
00031             { \
00032                 VECTOR_ENSURE_LENGTH_vec->capacity = 8; \
00033             } \
00034             else \
00035             { \
00036                 VECTOR_ENSURE_LENGTH_vec->capacity *= 2; \
00037             } \
00038             VECTOR_ENSURE_LENGTH_vec->data = realloc(VECTOR_ENSURE_LENGTH_vec->data, \
00039                                         VECTOR_ENSURE_LENGTH_vec->capacity * \
00040                                         sizeof(*VECTOR_ENSURE_LENGTH_vec->data)); /* NOLINT */ \
00041         } \
00042     } while (0)
00043 #define VECTOR_PUSH_BACK(VEC, VALUE) ({ \
00044     __auto_type VECTOR_PUSH_BACK_vec = VEC; \
00045     VECTOR_ENSURE_LENGTH(VEC, VECTOR_PUSH_BACK_vec->length); \
00046     VECTOR_PUSH_BACK_vec->data[VECTOR_PUSH_BACK_vec->length++] = VALUE; \
00047     VECTOR_PUSH_BACK_vec->length - 1; \
00048 })
00049
00050 #define VECTOR_PUSH_FRONT(VEC, VALUE) VECTOR_INSERT(VEC, 0, VALUE)
00051
00052 #define VECTOR_INSERT(VEC, IDX, VALUE) \
00053     do \
00054     { \
00055         __auto_type VECTOR_INSERT_vec = VEC; \
00056         size_t VECTOR_INSERT_index = IDX; \
00057         VECTOR_ENSURE_LENGTH(VEC, VECTOR_INSERT_vec->length); \
00058         for (size_t VECTOR_INSERT_i = VECTOR_INSERT_vec->length; VECTOR_INSERT_i > \
00059             VECTOR_INSERT_index; VECTOR_INSERT_i--) \
00060         { \
00061             VECTOR_INSERT_vec->data[VECTOR_INSERT_i] = VECTOR_INSERT_vec->data[VECTOR_INSERT_i - 1];

```

```

00061         }
00062         \
00063         VECTOR_INSERT_vec->length++;
00064     } while (0)
00065
00066 #define VECTOR_REMOVE(VEC, IDX)
00067     do
00068     {
00069         \
00070         __auto_type VECTOR_REMOVE_vec = VEC;
00071         \
00072         for (size_t VECTOR_REMOVE_i = (IDX); VECTOR_REMOVE_i < VECTOR_REMOVE_vec->length - 1;
00073             VECTOR_REMOVE_i++) \
00074         {
00075             \
00076             VECTOR_REMOVE_vec->data[VECTOR_REMOVE_i] = VECTOR_REMOVE_vec->data[VECTOR_REMOVE_i + 1];
00077         }
00078         \
00079         VECTOR_REMOVE_vec->length--;
00080     } while (0)
00081
00082 #define VECTOR_ITEM(VEC, IDX) ({ \
00083     size_t VECTOR_ITEM_idx = IDX; \
00084     __auto_type VECTOR_ITEM_vec = VEC; \
00085     __auto_type VECTOR_ITEM_result = (typeof(*VECTOR_ITEM_vec->data))VECTOR_INVALID_INDEX; \
00086     if (VECTOR_ITEM_idx < VECTOR_ITEM_vec->length) \
00087     { \
00088         VECTOR_ITEM_result = VECTOR_ITEM_vec->data[VECTOR_ITEM_idx]; \
00089     } \
00090     VECTOR_ITEM_result; \
00091 })
00092
00093 #define VECTOR_FIND(VEC, VALUE) ({ \
00094     __auto_type VECTOR_FIND_vec = VEC; \
00095     ssize_t VECTOR_FIND_result = VECTOR_INVALID_INDEX; \
00096     for (size_t VECTOR_FIND_i = 0; VECTOR_FIND_i < VECTOR_FIND_vec->length; VECTOR_FIND_i++) \
00097     {
00098         if (VECTOR_FIND_vec->data[VECTOR_FIND_i] == (VALUE)) \
00099         {
00100             VECTOR_FIND_result = VECTOR_FIND_i; \
00101             break; \
00102         } \
00103     } \
00104     VECTOR_FIND_result; \
00105 })
00106
00107 #define VECTOR_REMOVE_BY_VALUE(VEC, VALUE)
00108     do
00109     {
00110         \
00111         __auto_type VECTOR_REMOVE_BY_VALUE_vec = VEC;
00112         \
00113         __auto_type VECTOR_REMOVE_BY_VALUE_v = VALUE;
00114         \
00115         size_t VECTOR_REMOVE_BY_VALUE_i = VECTOR_FIND(VECTOR_REMOVE_BY_VALUE_vec,
00116             VECTOR_REMOVE_BY_VALUE_v); \
00117         VECTOR_REMOVE(VECTOR_REMOVE_BY_VALUE_vec, VECTOR_REMOVE_BY_VALUE_i);
00118     } while (0)
00119
00120 #define VECTOR_FOR_EACH(VEC, BINDING, ...)
00121     do
00122     {
00123         \
00124         __auto_type VECTOR_FOR_EACH_vec = VEC;
00125         \
00126         for (size_t VECTOR_FOR_EACH_i = 0; VECTOR_FOR_EACH_i < VECTOR_FOR_EACH_vec->length;
00127             VECTOR_FOR_EACH_i++) \
00128         {
00129             \
00130             __auto_type BINDING = &VECTOR_FOR_EACH_vec->data[VECTOR_FOR_EACH_i];
00131             \
00132             __VA_ARGS__
00133         }

```

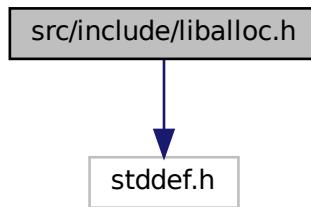
```

00120     } while (0)
00121
00122 #endif

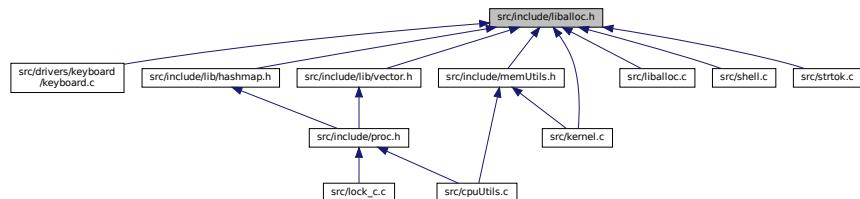
```

## 4.45 src/include/liballoc.h File Reference

#include <stddef.h>  
 Include dependency graph for liballoc.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [boundary\\_tag](#)

## Macros

- #define [NULL](#) 0

## Functions

- int [liballoc\\_lock](#) ()
- int [liballoc\\_unlock](#) ()
- void \* [liballoc\\_alloc](#) (int)
- int [liballoc\\_free](#) (void \*, int)
- void \* [malloc](#) (size\_t)
- void \* [realloc](#) (void \*, size\_t)
- void \* [calloc](#) (size\_t, size\_t)
- void [free](#) (void \*)

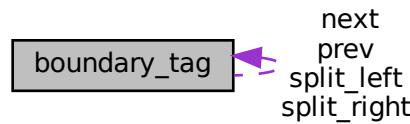
### 4.45.1 Data Structure Documentation

#### 4.45.1.1 struct boundary\_tag

This is a boundary tag which is prepended to the page or section of a page which we have allocated. It is used to identify valid memory blocks that the application is trying to free.

Definition at line 26 of file [liballoc.h](#).

Collaboration diagram for boundary\_tag:



#### Data Fields

int	index	
unsigned int	magic	
struct boundary_tag *	next	
struct boundary_tag *	prev	
unsigned int	real_size	
unsigned int	size	
struct boundary_tag *	split_left	
struct boundary_tag *	split_right	

### 4.45.2 Macro Definition Documentation

#### 4.45.2.1 NULL

```
#define NULL 0
```

Definition at line 11 of file [liballoc.h](#).

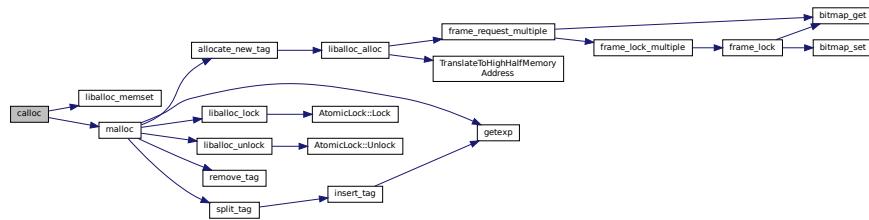
### 4.45.3 Function Documentation

#### 4.45.3.1 calloc()

```
void* calloc (
    size_t nobj,
    size_t size )
```

Definition at line 465 of file [liballoc.c](#).

Here is the call graph for this function:

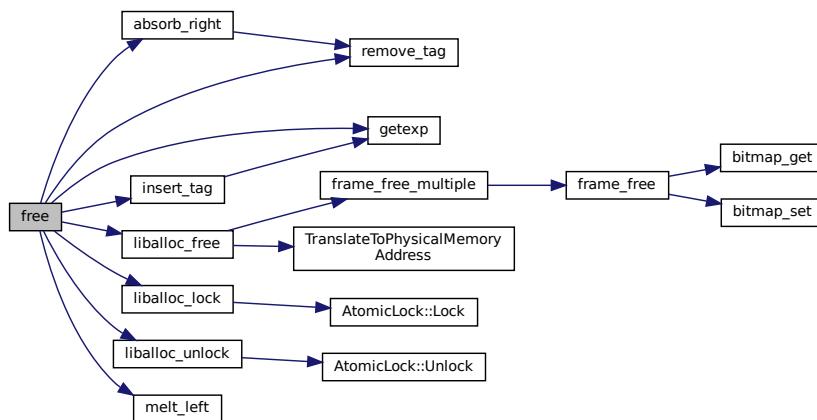


#### 4.45.3.2 free()

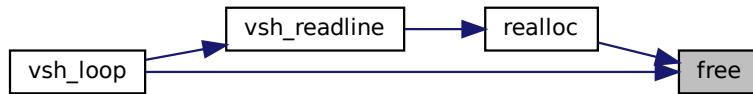
```
void free (
    void * ptr )
```

Definition at line 377 of file [liballoc.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.45.3.3 liballoc\_alloc()

```
void* liballoc_alloc (
    int    )
```

This is the hook into the local system which allocates pages. It accepts an integer parameter which is the number of pages required. The page size was set up in the liballoc\_init function.

##### Returns

NULL if the pages were not allocated.  
A pointer to the allocated memory.

#### 4.45.3.4 liballoc\_free()

```
int liballoc_free (
    void * ,
    int    )
```

This frees previously allocated memory. The void\* parameter passed to the function is the exact same value returned from a previous liballoc\_alloc call.

The integer value is the number of pages to free.

##### Returns

0 if the memory was successfully freed.

#### 4.45.3.5 liballoc\_lock()

```
int liballoc_lock ( )
```

This function is supposed to lock the memory data structures. It could be as simple as disabling interrupts or acquiring a spinlock. It's up to you to decide.

##### Returns

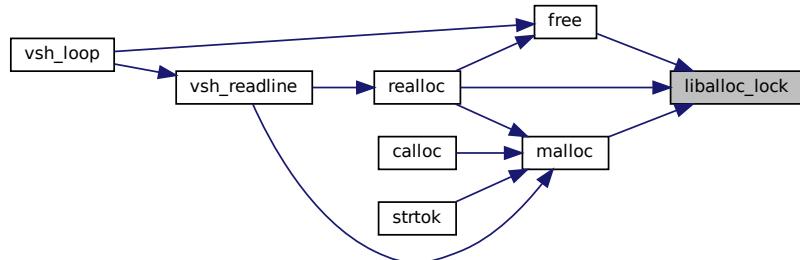
0 if the lock was acquired successfully. Anything else is failure.

Definition at line 10 of file [alloc.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.45.3.6 liballoc\_unlock()

```
int liballoc_unlock ( )
```

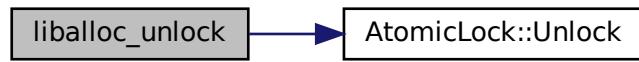
This function unlocks what was previously locked by the liballoc\_lock function. If it disabled interrupts, it enables interrupts. If it had acquired a spinlock, it releases the spinlock. etc.

**Returns**

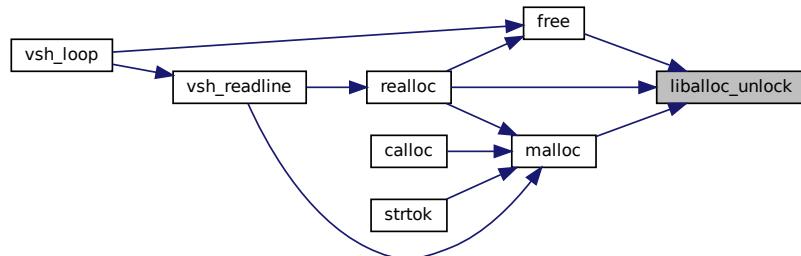
0 if the lock was successfully released.

Definition at line 18 of file [alloc.cpp](#).

Here is the call graph for this function:



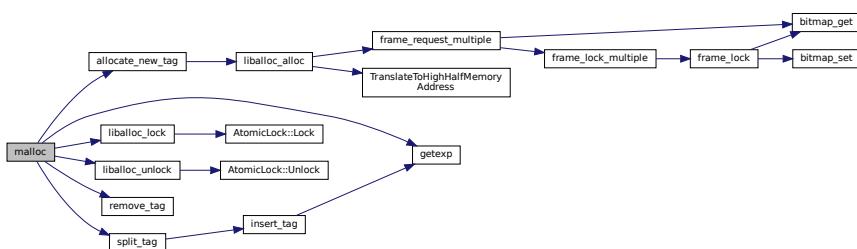
Here is the caller graph for this function:

**4.45.3.7 malloc()**

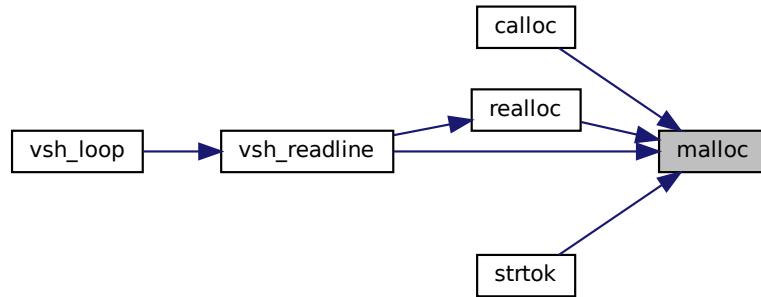
```
void* malloc (
    size_t size )
```

Definition at line 273 of file [liballoc.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

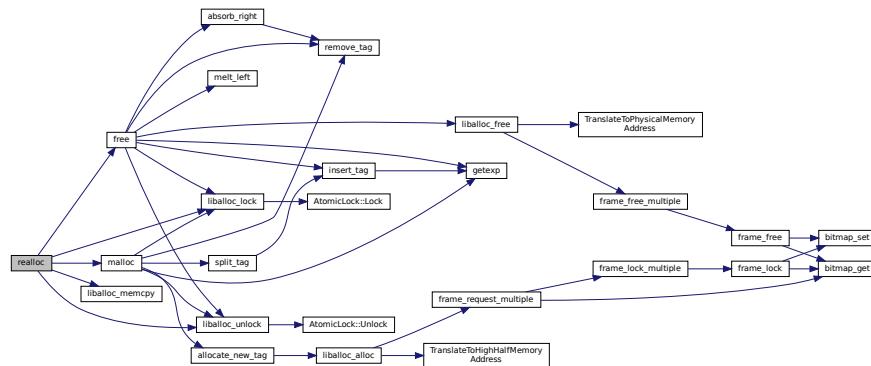


#### 4.45.3.8 realloc()

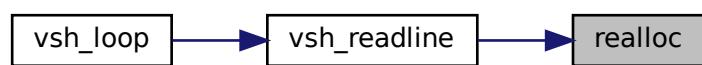
```
void* realloc (
    void * p,
    size_t size )
```

Definition at line 479 of file [liballoc.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.46 liballoc.h

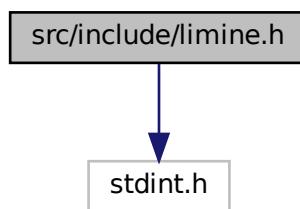
```

00001 #ifndef _LIBALLOC_H
00002 #define _LIBALLOC_H
00003
00004 // If we are told to not define our own size_t, then we
00005 // skip the define.
00006 #ifndef _ALLOC_SKIP_DEFINE
00007
00008 #include <stddef.h>
00009
00010 #ifndef NULL
00011 #define NULL 0
00012 #endif
00013
00014 #endif
00015
00016 #ifdef __cplusplus
00017 extern "C"
00018 {
00019 #endif
00020
00021     struct boundary_tag
00022     {
00023         unsigned int magic;      //< It's a kind of ...
00024         unsigned int size;       //< Requested size.
00025         unsigned int real_size; //< Actual size.
00026         int index;             //< Location in the page table.
00027
00028         struct boundary_tag *split_left; //< Linked-list info for broken pages.
00029         struct boundary_tag *split_right; //< The same.
00030
00031         struct boundary_tag *next; //< Linked list info.
00032         struct boundary_tag *prev; //< Linked list info.
00033     };
00034
00035     extern int liballoc_lock();
00036
00037     extern int liballoc_unlock();
00038
00039     extern void *liballoc_alloc(int);
00040
00041     extern int liballoc_free(void *, int);
00042
00043     void *malloc(size_t);           //< The standard function.
00044     void *realloc(void *, size_t); //< The standard function.
00045     void *calloc(size_t, size_t); //< The standard function.
00046     void free(void *);           //< The standard function.
00047
00048 #endif //__cplusplus
00049 }
00050 #endif
00051
00052 #endif

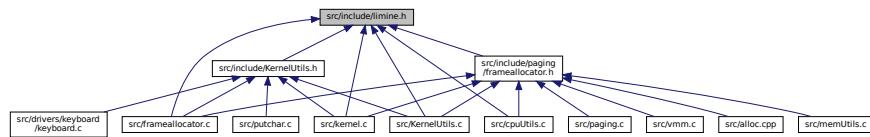
```

## 4.47 src/include/limine.h File Reference

```
#include <stdint.h>
Include dependency graph for limine.h:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `limine_uuid`
- struct `limine_file`
- struct `limine_bootloader_info_response`
- struct `limine_bootloader_info_request`
- struct `limine_stack_size_response`
- struct `limine_stack_size_request`
- struct `limine_hhdm_response`
- struct `limine_hhdm_request`
- struct `limine_FRAMEBUFFER`
- struct `limine_FRAMEBUFFER_response`
- struct `limine_FRAMEBUFFER_request`
- struct `limine_terminal`
- struct `limine_terminal_response`
- struct `limine_terminal_request`
- struct `limine_5_level_paging_response`
- struct `limine_5_level_paging_request`
- struct `limine_smp_request`
- struct `limine_memmap_entry`
- struct `limine_memmap_response`
- struct `limine_memmap_request`
- struct `limine_entry_point_response`
- struct `limine_entry_point_request`
- struct `limine_kernel_file_response`
- struct `limine_kernel_file_request`
- struct `limine_module_response`
- struct `limine_module_request`
- struct `limine_rsdp_response`
- struct `limine_rsdp_request`
- struct `limine_smbios_response`
- struct `limine_smbios_request`
- struct `limine_efi_system_table_response`
- struct `limine_efi_system_table_request`
- struct `limine_boot_time_response`
- struct `limine_boot_time_request`
- struct `limine_kernel_address_response`
- struct `limine_kernel_address_request`
- struct `limine_dtb_response`
- struct `limine_dtb_request`

## Macros

- `#define LIMINE_PTR(TYPE) TYPE`
- `#define LIMINE_COMMON_MAGIC 0xc7b1dd30df4c8b88, 0x0a82e883a194f07b`
- `#define LIMINE_MEDIA_TYPE_GENERIC 0`
- `#define LIMINE_MEDIA_TYPE_OPTICAL 1`
- `#define LIMINE_MEDIA_TYPE_TFTP 2`
- `#define LIMINE_BOOTLOADER_INFO_REQUEST { LIMINE_COMMON_MAGIC, 0xf55038d8e2a1202f, 0x279426fcf5f59740 }`
- `#define LIMINE_STACK_SIZE_REQUEST { LIMINE_COMMON_MAGIC, 0x224ef0460a8e8926, 0xe1cb0fc25f46ea3d }`
- `#define LIMINE_HHDM_REQUEST { LIMINE_COMMON_MAGIC, 0x48dcf1cb8ad2b852, 0x63984e959a98244b }`
- `#define LIMINE_FRAMEBUFFER_REQUEST { LIMINE_COMMON_MAGIC, 0x9d5827dcd881dd75, 0xa3148604f6fab11b }`
- `#define LIMINE_FRAMEBUFFER_RGB 1`
- `#define LIMINE_TERMINAL_REQUEST { LIMINE_COMMON_MAGIC, 0xc8ac59310c2b0844, 0xa68d0c7265d38878 }`
- `#define LIMINE_TERMINAL_CB_DEC 10`
- `#define LIMINE_TERMINAL_CB_BELL 20`
- `#define LIMINE_TERMINAL_CB_PRIVATE_ID 30`
- `#define LIMINE_TERMINAL_CB_STATUS_REPORT 40`
- `#define LIMINE_TERMINAL_CB_POS_REPORT 50`
- `#define LIMINE_TERMINAL_CB_KBD_LEDS 60`
- `#define LIMINE_TERMINAL_CB_MODE 70`
- `#define LIMINE_TERMINAL_CB_LINUX 80`
- `#define LIMINE_TERMINAL_CTX_SIZE ((uint64_t)(-1))`
- `#define LIMINE_TERMINAL_CTX_SAVE ((uint64_t)(-2))`
- `#define LIMINE_TERMINAL_CTX_RESTORE ((uint64_t)(-3))`
- `#define LIMINE_TERMINAL_FULL_REFRESH ((uint64_t)(-4))`
- `#define LIMINE_5_LEVEL_PAGING_REQUEST { LIMINE_COMMON_MAGIC, 0x94469551da9b3192, 0xebbe5e86db7382888 }`
- `#define LIMINE_SMP_REQUEST { LIMINE_COMMON_MAGIC, 0x95a67b819a1b857e, 0xa0b61b723b6a73e0 }`
- `#define LIMINE_MEMMAP_REQUEST { LIMINE_COMMON_MAGIC, 0x67cf3d9d378a806f, 0xe304acd50c3c62 }`
- `#define LIMINE_MEMMAP_USABLE 0`
- `#define LIMINE_MEMMAP_RESERVED 1`
- `#define LIMINE_MEMMAP_ACPI_RECLAMABLE 2`
- `#define LIMINE_MEMMAP_ACPI_NVS 3`
- `#define LIMINE_MEMMAP_BAD_MEMORY 4`
- `#define LIMINE_MEMMAP_BOOTLOADER_RECLAMABLE 5`
- `#define LIMINE_MEMMAP_KERNEL_AND_MODULES 6`
- `#define LIMINE_MEMMAP_FRAMEBUFFER 7`
- `#define LIMINE_ENTRY_POINT_REQUEST { LIMINE_COMMON_MAGIC, 0x13d86c035a1cd3e1, 0x2b0caa89d8f3026a }`
- `#define LIMINE_KERNEL_FILE_REQUEST { LIMINE_COMMON_MAGIC, 0xad97e90e83f1ed67, 0x31eb5d1c5ff23b69 }`
- `#define LIMINE_MODULE_REQUEST { LIMINE_COMMON_MAGIC, 0x3e7e279702be32af, 0xca1c4f3bd1280cee }`
- `#define LIMINE_RSDP_REQUEST { LIMINE_COMMON_MAGIC, 0xc5e77b6b397e7b43, 0x27637845accdf3c }`
- `#define LIMINE_SMBIOS_REQUEST { LIMINE_COMMON_MAGIC, 0x9e9046f11e095391, 0xaa4a520fefbde5ee }`
- `#define LIMINE_EFI_SYSTEM_TABLE_REQUEST { LIMINE_COMMON_MAGIC, 0x5ceba5163eaaf6d6, 0x0a6981610cf65fcc }`

- #define LIMINE\_BOOT\_TIME\_REQUEST { LIMINE\_COMMON\_MAGIC, 0x502746e184c088aa, 0xfc5ec83e6327893 }
- #define LIMINE\_KERNEL\_ADDRESS\_REQUEST { LIMINE\_COMMON\_MAGIC, 0x71ba76863cc55f63, 0xb2644a48c516a487 }
- #define LIMINE\_DTB\_REQUEST { LIMINE\_COMMON\_MAGIC, 0xb40ddb48fb54bac7, 0x545081493f81ffb7 }

## Typedefs

- typedef void(\* limine\_terminal\_write) (struct limine\_terminal \*, const char \*, uint64\_t)
- typedef void(\* limine\_terminal\_callback) (struct limine\_terminal \*, uint64\_t, uint64\_t, uint64\_t, uint64\_t)
- typedef void(\* limine\_goto\_address) (struct limine\_smp\_info \*)
- typedef void(\* limine\_entry\_point) (void)

### 4.47.1 Data Structure Documentation

#### 4.47.1.1 struct limine\_uuid

Definition at line 20 of file [limine.h](#).

##### Data Fields

uint32_t	a	
uint16_t	b	
uint16_t	c	
uint8_t	d[8]	

#### 4.47.1.2 struct limine\_stack\_size\_response

Definition at line 68 of file [limine.h](#).

##### Data Fields

uint64_t	revision	
----------	----------	--

#### 4.47.1.3 struct limine\_hhdm\_response

Definition at line 83 of file [limine.h](#).

##### Data Fields

uint64_t	offset	
uint64_t	revision	

#### 4.47.1.4 struct limine\_5\_level\_paging\_response

Definition at line 177 of file [limine.h](#).

Data Fields

uint64_t	revision	
----------	----------	--

#### 4.47.1.5 struct limine\_memmap\_entry

Definition at line 258 of file [limine.h](#).

Data Fields

uint64_t	base	
uint64_t	length	
uint64_t	type	

#### 4.47.1.6 struct limine\_entry\_point\_response

Definition at line 282 of file [limine.h](#).

Data Fields

uint64_t	revision	
----------	----------	--

#### 4.47.1.7 struct limine\_boot\_time\_response

Definition at line 374 of file [limine.h](#).

Data Fields

int64_t	boot_time	
uint64_t	revision	

#### 4.47.1.8 struct limine\_kernel\_address\_response

Definition at line 389 of file [limine.h](#).

Data Fields

uint64_t	physical_base	
uint64_t	revision	
uint64_t	virtual_base	

## 4.47.2 Macro Definition Documentation

### 4.47.2.1 LIMINE\_5\_LEVEL\_PAGING\_REQUEST

```
#define LIMINE_5_LEVEL_PAGING_REQUEST { LIMINE_COMMON_MAGIC, 0x94469551da9b3192, 0xebbe5e86db7382888 }
```

Definition at line 175 of file [limine.h](#).

### 4.47.2.2 LIMINE\_BOOT\_TIME\_REQUEST

```
#define LIMINE_BOOT_TIME_REQUEST { LIMINE_COMMON_MAGIC, 0x502746e184c088aa, 0xfbcb5ec83e6327893 }
```

Definition at line 372 of file [limine.h](#).

### 4.47.2.3 LIMINE\_BOOTLOADER\_INFO\_REQUEST

```
#define LIMINE_BOOTLOADER_INFO_REQUEST { LIMINE_COMMON_MAGIC, 0xf55038d8e2a1202f, 0x279426fcf5f59740 }
```

Definition at line 50 of file [limine.h](#).

### 4.47.2.4 LIMINE\_COMMON\_MAGIC

```
#define LIMINE_COMMON_MAGIC 0xc7b1dd30df4c8b88, 0x0a82e883a194f07b
```

Definition at line 18 of file [limine.h](#).

### 4.47.2.5 LIMINE\_DTB\_REQUEST

```
#define LIMINE_DTB_REQUEST { LIMINE_COMMON_MAGIC, 0xb40ddb48fb54bac7, 0x545081493f81ffb7 }
```

Definition at line 403 of file [limine.h](#).

#### 4.47.2.6 LIMINE\_EFI\_SYSTEM\_TABLE\_REQUEST

```
#define LIMINE_EFI_SYSTEM_TABLE_REQUEST { LIMINE_COMMON_MAGIC, 0x5ceba5163eaaf6d6, 0x0a6981610cf65fcc }
```

Definition at line 357 of file [limine.h](#).

#### 4.47.2.7 LIMINE\_ENTRY\_POINT\_REQUEST

```
#define LIMINE_ENTRY_POINT_REQUEST { LIMINE_COMMON_MAGIC, 0x13d86c035a1cd3e1, 0x2b0caa89d8f3026a }
```

Definition at line 278 of file [limine.h](#).

#### 4.47.2.8 LIMINE\_FRAMEBUFFER\_REQUEST

```
#define LIMINE_FRAMEBUFFER_REQUEST { LIMINE_COMMON_MAGIC, 0x9d5827dcd881dd75, 0xa3148604f6fab11b }
```

Definition at line 96 of file [limine.h](#).

#### 4.47.2.9 LIMINE\_FRAMEBUFFER\_RGB

```
#define LIMINE_FRAMEBUFFER_RGB 1
```

Definition at line 98 of file [limine.h](#).

#### 4.47.2.10 LIMINE\_HHDM\_REQUEST

```
#define LIMINE_HHDM_REQUEST { LIMINE_COMMON_MAGIC, 0x48dcf1cb8ad2b852, 0x63984e959a98244b }
```

Definition at line 81 of file [limine.h](#).

#### 4.47.2.11 LIMINE\_KERNEL\_ADDRESS\_REQUEST

```
#define LIMINE_KERNEL_ADDRESS_REQUEST { LIMINE_COMMON_MAGIC, 0x71ba76863cc55f63, 0xb2644a48c516a487 }
```

Definition at line 387 of file [limine.h](#).

#### 4.47.2.12 LIMINE\_KERNEL\_FILE\_REQUEST

```
#define LIMINE_KERNEL_FILE_REQUEST { LIMINE_COMMON_MAGIC, 0xad97e90e83f1ed67, 0x31eb5d1c5ff23b69 }
```

Definition at line 295 of file [limine.h](#).

#### 4.47.2.13 LIMINE\_MEDIA\_TYPE\_GENERIC

```
#define LIMINE_MEDIA_TYPE_GENERIC 0
```

Definition at line 27 of file [limine.h](#).

#### 4.47.2.14 LIMINE\_MEDIA\_TYPE\_OPTICAL

```
#define LIMINE_MEDIA_TYPE_OPTICAL 1
```

Definition at line 28 of file [limine.h](#).

#### 4.47.2.15 LIMINE\_MEDIA\_TYPE\_TFTP

```
#define LIMINE_MEDIA_TYPE_TFTP 2
```

Definition at line 29 of file [limine.h](#).

#### 4.47.2.16 LIMINE\_MEMMAP\_ACPI\_NVS

```
#define LIMINE_MEMMAP_ACPI_NVS 3
```

Definition at line 252 of file [limine.h](#).

#### 4.47.2.17 LIMINE\_MEMMAP\_ACPI\_RECLAIMABLE

```
#define LIMINE_MEMMAP_ACPI_RECLAIMABLE 2
```

Definition at line 251 of file [limine.h](#).

#### 4.47.2.18 LIMINE\_MEMMAP\_BAD\_MEMORY

```
#define LIMINE_MEMMAP_BAD_MEMORY 4
```

Definition at line 253 of file [limine.h](#).

#### 4.47.2.19 LIMINE\_MEMMAP\_BOOTLOADER\_RECLAIMABLE

```
#define LIMINE_MEMMAP_BOOTLOADER_RECLAIMABLE 5
```

Definition at line 254 of file [limine.h](#).

#### 4.47.2.20 LIMINE\_MEMMAP\_FRAMEBUFFER

```
#define LIMINE_MEMMAP_FRAMEBUFFER 7
```

Definition at line 256 of file [limine.h](#).

#### 4.47.2.21 LIMINE\_MEMMAP\_KERNEL\_AND\_MODULES

```
#define LIMINE_MEMMAP_KERNEL_AND_MODULES 6
```

Definition at line 255 of file [limine.h](#).

#### 4.47.2.22 LIMINE\_MEMMAP\_REQUEST

```
#define LIMINE_MEMMAP_REQUEST { LIMINE_COMMON_MAGIC, 0x67cf3d9d378a806f, 0xe304acdfc50c3c62 }
```

Definition at line 247 of file [limine.h](#).

#### 4.47.2.23 LIMINE\_MEMMAP\_RESERVED

```
#define LIMINE_MEMMAP_RESERVED 1
```

Definition at line 250 of file [limine.h](#).

#### 4.47.2.24 LIMINE\_MEMMAP\_USABLE

```
#define LIMINE_MEMMAP_USABLE 0
```

Definition at line 249 of file [limine.h](#).

#### 4.47.2.25 LIMINE\_MODULE\_REQUEST

```
#define LIMINE_MODULE_REQUEST { LIMINE_COMMON_MAGIC, 0x3e7e279702be32af, 0xca1c4f3bd1280cee }
```

Definition at line 310 of file [limine.h](#).

#### 4.47.2.26 LIMINE\_PTR

```
#define LIMINE_PTR(
```

```
    TYPE ) TYPE
```

Definition at line 15 of file [limine.h](#).

#### 4.47.2.27 LIMINE\_RSDP\_REQUEST

```
#define LIMINE_RSDP_REQUEST { LIMINE_COMMON_MAGIC, 0xc5e77b6b397e7b43, 0x27637845accdf3c }
```

Definition at line 326 of file [limine.h](#).

#### 4.47.2.28 LIMINE\_SMBIOS\_REQUEST

```
#define LIMINE_SMBIOS_REQUEST { LIMINE_COMMON_MAGIC, 0x9e9046f11e095391, 0xaa4a520fefbde5ee }
```

Definition at line 341 of file [limine.h](#).

#### 4.47.2.29 LIMINE\_SMP\_REQUEST

```
#define LIMINE_SMP_REQUEST { LIMINE_COMMON_MAGIC, 0x95a67b819a1b857e, 0xa0b61b723b6a73e0 }
```

Definition at line 189 of file [limine.h](#).

#### 4.47.2.30 LIMINE\_STACK\_SIZE\_REQUEST

```
#define LIMINE_STACK_SIZE_REQUEST { LIMINE_COMMON_MAGIC, 0x224ef0460a8e8926, 0xe1cb0fc25f46ea3d }
```

Definition at line 66 of file [limine.h](#).

#### 4.47.2.31 LIMINE\_TERMINAL\_CB\_BELL

```
#define LIMINE_TERMINAL_CB_BELL 20
```

Definition at line 135 of file [limine.h](#).

#### 4.47.2.32 LIMINE\_TERMINAL\_CB\_DEC

```
#define LIMINE_TERMINAL_CB_DEC 10
```

Definition at line 134 of file [limine.h](#).

#### 4.47.2.33 LIMINE\_TERMINAL\_CB\_KBD\_LEDS

```
#define LIMINE_TERMINAL_CB_KBD_LEDS 60
```

Definition at line 139 of file [limine.h](#).

#### 4.47.2.34 LIMINE\_TERMINAL\_CB\_LINUX

```
#define LIMINE_TERMINAL_CB_LINUX 80
```

Definition at line 141 of file [limine.h](#).

#### 4.47.2.35 LIMINE\_TERMINAL\_CB\_MODE

```
#define LIMINE_TERMINAL_CB_MODE 70
```

Definition at line 140 of file [limine.h](#).

#### 4.47.2.36 LIMINE\_TERMINAL\_CB\_POS\_REPORT

```
#define LIMINE_TERMINAL_CB_POS_REPORT 50
```

Definition at line 138 of file [limine.h](#).

#### 4.47.2.37 LIMINE\_TERMINAL\_CB\_PRIVATE\_ID

```
#define LIMINE_TERMINAL_CB_PRIVATE_ID 30
```

Definition at line 136 of file [limine.h](#).

#### 4.47.2.38 LIMINE\_TERMINAL\_CB\_STATUS\_REPORT

```
#define LIMINE_TERMINAL_CB_STATUS_REPORT 40
```

Definition at line 137 of file [limine.h](#).

#### 4.47.2.39 LIMINE\_TERMINAL\_CTX\_RESTORE

```
#define LIMINE_TERMINAL_CTX_RESTORE ((uint64_t)(-3))
```

Definition at line 145 of file [limine.h](#).

#### 4.47.2.40 LIMINE\_TERMINAL\_CTX\_SAVE

```
#define LIMINE_TERMINAL_CTX_SAVE ((uint64_t)(-2))
```

Definition at line 144 of file [limine.h](#).

#### 4.47.2.41 LIMINE\_TERMINAL\_CTX\_SIZE

```
#define LIMINE_TERMINAL_CTX_SIZE ((uint64_t)(-1))
```

Definition at line 143 of file [limine.h](#).

#### 4.47.2.42 LIMINE\_TERMINAL\_FULL\_REFRESH

```
#define LIMINE_TERMINAL_FULL_REFRESH ((uint64_t)(-4))
```

Definition at line 146 of file [limine.h](#).

#### 4.47.2.43 LIMINE\_TERMINAL\_REQUEST

```
#define LIMINE_TERMINAL_REQUEST { LIMINE_COMMON_MAGIC, 0xc8ac59310c2b0844, 0xa68d0c7265d38878 }
```

Definition at line 132 of file [limine.h](#).

### 4.47.3 Typedef Documentation

#### 4.47.3.1 limine\_entry\_point

```
typedef void(* limine_entry_point) (void)
```

Definition at line 280 of file [limine.h](#).

#### 4.47.3.2 limine\_goto\_address

```
typedef void(* limine_goto_address) (struct limine_smp_info *)
```

Definition at line 193 of file [limine.h](#).

#### 4.47.3.3 limine\_terminal\_callback

```
typedef void(* limine_terminal_callback) (struct limine_terminal *, uint64_t, uint64_t, uint64_t, uint64_t)
```

Definition at line 151 of file [limine.h](#).

#### 4.47.3.4 limine\_terminal\_write

```
typedef void(* limine_terminal_write) (struct limine_terminal *, const char *, uint64_t)
```

Definition at line 150 of file [limine.h](#).

## 4.48 limine.h

```

00001 #ifndef _LIMINE_H
00002 #define _LIMINE_H 1
00003
00004 #ifdef __cplusplus
00005 extern "C" {
00006 #endif
00007
00008 #include <stdint.h>
00009
00010 /* Misc */
00011
00012 #ifdef LIMINE_NO_POINTERS
00013 # define LIMINE_PTR(TYPE) uint64_t
00014 #else
00015 # define LIMINE_PTR(TYPE) TYPE
00016 #endif
00017
00018 #define LIMINE_COMMON_MAGIC 0xc7b1dd30df4c8b88, 0x0a82e883a194f07b
00019
00020 struct limine_uuid {
00021     uint32_t a;
00022     uint16_t b;
00023     uint16_t c;
00024     uint8_t d[8];
00025 };
00026
00027 #define LIMINE_MEDIA_TYPE_GENERIC 0
00028 #define LIMINE_MEDIA_TYPE_OPTICAL 1
00029 #define LIMINE_MEDIA_TYPE_TFTP 2
00030
00031 struct limine_file {
00032     uint64_t revision;
00033     LIMINE_PTR(void *) address;
00034     uint64_t size;
00035     LIMINE_PTR(char *) path;
00036     LIMINE_PTR(char *) cmdline;
00037     uint32_t media_type;
00038     uint32_t unused;
00039     uint32_t tftp_ip;
00040     uint32_t tftp_port;
00041     uint32_t partition_index;
00042     uint32_t mbr_disk_id;
00043     struct limine_uuid gpt_disk_uuid;
00044     struct limine_uuid gpt_part_uuid;
00045     struct limine_uuid part_uuid;
00046 };
00047
00048 /* Boot info */
00049
00050 #define LIMINE_BOOTLOADER_INFO_REQUEST { LIMINE_COMMON_MAGIC, 0xf55038d8e2a1202f, 0x279426fcf5f59740 }
00051
00052 struct limine_bootloader_info_response {
00053     uint64_t revision;
00054     LIMINE_PTR(char *) name;
00055     LIMINE_PTR(char *) version;
00056 };
00057
00058 struct limine_bootloader_info_request {
00059     uint64_t id[4];
00060     uint64_t revision;
00061     LIMINE_PTR(struct limine_bootloader_info_response *) response;
00062 };
00063
00064 /* Stack size */
00065
00066 #define LIMINE_STACK_SIZE_REQUEST { LIMINE_COMMON_MAGIC, 0x224ef0460a8e8926, 0xe1cb0fc25f46ea3d }
00067
00068 struct limine_stack_size_response {
00069     uint64_t revision;
00070 };
00071
00072 struct limine_stack_size_request {
00073     uint64_t id[4];
00074     uint64_t revision;
00075     LIMINE_PTR(struct limine_stack_size_response *) response;
00076     uint64_t stack_size;
00077 };
00078
00079 /* HHDM */
00080
00081 #define LIMINE_HHDM_REQUEST { LIMINE_COMMON_MAGIC, 0x48dcf1cb8ad2b852, 0x63984e959a98244b }
00082
00083 struct limine_hhdm_response {
00084     uint64_t revision;
00085     uint64_t offset;

```

```
00086 };
00087
00088 struct limine_hhdm_request {
00089     uint64_t id[4];
00090     uint64_t revision;
00091     LIMINE_PTR(struct limine_hhdm_response *) response;
00092 };
00093
00094 /* Framebuffer */
00095
00096 #define LIMINE_FRAMEBUFFER_REQUEST { LIMINE_COMMON_MAGIC, 0x9d5827dcd881dd75, 0xa3148604f6fab11b }
00097
00098 #define LIMINE_FRAMEBUFFER_RGB 1
00099
00100 struct limine_framebuffer {
00101     LIMINE_PTR(void *) address;
00102     uint64_t width;
00103     uint64_t height;
00104     uint64_t pitch;
00105     uint16_t bpp;
00106     uint8_t memory_model;
00107     uint8_t red_mask_size;
00108     uint8_t red_mask_shift;
00109     uint8_t green_mask_size;
00110     uint8_t green_mask_shift;
00111     uint8_t blue_mask_size;
00112     uint8_t blue_mask_shift;
00113     uint8_t unused[7];
00114     uint64_t edid_size;
00115     LIMINE_PTR(void *) edid;
00116 };
00117
00118 struct limine_framebuffer_response {
00119     uint64_t revision;
00120     uint64_t framebuffer_count;
00121     LIMINE_PTR(struct limine_framebuffer **) framebuffers;
00122 };
00123
00124 struct limine_framebuffer_request {
00125     uint64_t id[4];
00126     uint64_t revision;
00127     LIMINE_PTR(struct limine_framebuffer_response *) response;
00128 };
00129
00130 /* Terminal */
00131
00132 #define LIMINE_TERMINAL_REQUEST { LIMINE_COMMON_MAGIC, 0xc8ac59310c2b0844, 0xa68d0c7265d38878 }
00133
00134 #define LIMINE_TERMINAL_CB_DEC 10
00135 #define LIMINE_TERMINAL_CB_BELL 20
00136 #define LIMINE_TERMINAL_CB_PRIVATE_ID 30
00137 #define LIMINE_TERMINAL_CB_STATUS_REPORT 40
00138 #define LIMINE_TERMINAL_CB_POS_REPORT 50
00139 #define LIMINE_TERMINAL_CB_KBD_LEDS 60
00140 #define LIMINE_TERMINAL_CB_MODE 70
00141 #define LIMINE_TERMINAL_CB_LINUX 80
00142
00143 #define LIMINE_TERMINAL_CTX_SIZE ((uint64_t)(-1))
00144 #define LIMINE_TERMINAL_CTX_SAVE ((uint64_t)(-2))
00145 #define LIMINE_TERMINAL_CTX_RESTORE ((uint64_t)(-3))
00146 #define LIMINE_TERMINAL_FULL_REFRESH ((uint64_t)(-4))
00147
00148 struct limine_terminal;
00149
00150 typedef void (*limine_terminal_write)(struct limine_terminal *, const char *, uint64_t);
00151 typedef void (*limine_terminal_callback)(struct limine_terminal *, uint64_t, uint64_t, uint64_t,
00152                                         uint64_t);
00153
00153 struct limine_terminal {
00154     uint64_t columns;
00155     uint64_t rows;
00156     LIMINE_PTR(struct limine_framebuffer *) framebuffer;
00157 };
00158
00159 struct limine_terminal_response {
00160     uint64_t revision;
00161     uint64_t terminal_count;
00162     LIMINE_PTR(struct limine_terminal **) terminals;
00163     LIMINE_PTR(limine_terminal_write) write;
00164 };
00165
00166 struct limine_terminal_request {
00167     uint64_t id[4];
00168     uint64_t revision;
00169     LIMINE_PTR(struct limine_terminal_response *) response;
00170     LIMINE_PTR(limine_terminal_callback) callback;
00171 };
```

```

00172 /* 5-level paging */
00173 #define LIMINE_5_LEVEL_PAGING_REQUEST { LIMINE_COMMON_MAGIC, 0x94469551da9b3192, 0xebe5e86db7382888 }
00174
00175 struct limine_5_level.paging_response {
00176     uint64_t revision;
00177 };
00178
00179 struct limine_5_level.paging_request {
00180     uint64_t id[4];
00181     uint64_t revision;
00182     LIMINE_PTR(struct limine_5_level.paging_response *) response;
00183 };
00184
00185 */
00186
00187 /* SMP */
00188
00189 #define LIMINE_SMP_REQUEST { LIMINE_COMMON_MAGIC, 0x95a67b819a1b857e, 0xa0b61b723b6a73e0 }
00190
00191 struct limine_smp_info;
00192
00193 typedef void (*limine_goto_address)(struct limine_smp_info *);
00194
00195 #if defined (__x86_64__) || defined (__i386__)
00196
00197 #define LIMINE_SMP_X2APIC (1 << 0)
00198
00199 struct limine_smp_info {
00200     uint32_t processor_id;
00201     uint32_t lapic_id;
00202     uint64_t reserved;
00203     LIMINE_PTR(limine_goto_address) goto_address;
00204     uint64_t extra_argument;
00205 };
00206
00207 struct limine_smp_response {
00208     uint64_t revision;
00209     uint32_t flags;
00210     uint32_t bsp_lapic_id;
00211     uint64_t cpu_count;
00212     LIMINE_PTR(struct limine_smp_info **) cpus;
00213 };
00214
00215 #elif defined (__aarch64__)
00216
00217 struct limine_smp_info {
00218     uint32_t processor_id;
00219     uint32_t gic_iface_no;
00220     uint64_t mpidr;
00221     uint64_t reserved;
00222     LIMINE_PTR(limine_goto_address) goto_address;
00223     uint64_t extra_argument;
00224 };
00225
00226 struct limine_smp_response {
00227     uint64_t revision;
00228     uint32_t flags;
00229     uint64_t bsp_mpidr;
00230     uint64_t cpu_count;
00231     LIMINE_PTR(struct limine_smp_info **) cpus;
00232 };
00233
00234 #else
00235 #error Unknown architecture
00236 #endif
00237
00238 struct limine_smp_request {
00239     uint64_t id[4];
00240     uint64_t revision;
00241     LIMINE_PTR(struct limine_smp_response *) response;
00242     uint64_t flags;
00243 };
00244
00245 /* Memory map */
00246
00247 #define LIMINE_MEMMAP_REQUEST { LIMINE_COMMON_MAGIC, 0x67cf3d9d378a806f, 0xe304acdfc50c3c62 }
00248
00249 #define LIMINE_MEMMAP_USABLE          0
00250 #define LIMINE_MEMMAP_RESERVED        1
00251 #define LIMINE_MEMMAP_ACPI_RECLAIMABLE 2
00252 #define LIMINE_MEMMAP_ACPI_NVS        3
00253 #define LIMINE_MEMMAP_BAD_MEMORY      4
00254 #define LIMINE_MEMMAP_BOOTLOADER_RECLAIMABLE 5
00255 #define LIMINE_MEMMAP_KERNEL_AND_MODULES 6
00256 #define LIMINE_MEMMAP_FRAMEBUFFER       7
00257
00258 struct limine_memmap_entry {

```

```
00259     uint64_t base;
00260     uint64_t length;
00261     uint64_t type;
00262 };
00263
00264 struct limine_memmap_response {
00265     uint64_t revision;
00266     uint64_t entry_count;
00267     LIMINE_PTR(struct limine_memmap_entry **) entries;
00268 };
00269
00270 struct limine_memmap_request {
00271     uint64_t id[4];
00272     uint64_t revision;
00273     LIMINE_PTR(struct limine_memmap_response *) response;
00274 };
00275
00276 /* Entry point */
00277
00278 #define LIMINE_ENTRY_POINT_REQUEST { LIMINE_COMMON_MAGIC, 0x13d86c035a1cd3e1, 0x2b0caa89d8f3026a }
00279
00280 typedef void (*limine_entry_point)(void);
00281
00282 struct limine_entry_point_response {
00283     uint64_t revision;
00284 };
00285
00286 struct limine_entry_point_request {
00287     uint64_t id[4];
00288     uint64_t revision;
00289     LIMINE_PTR(struct limine_entry_point_response *) response;
00290     LIMINE_PTR(limine_entry_point) entry;
00291 };
00292
00293 /* Kernel File */
00294
00295 #define LIMINE_KERNEL_FILE_REQUEST { LIMINE_COMMON_MAGIC, 0xad97e90e83fled67, 0x31eb5d1c5ff23b69 }
00296
00297 struct limine_kernel_file_response {
00298     uint64_t revision;
00299     LIMINE_PTR(struct limine_file *) kernel_file;
00300 };
00301
00302 struct limine_kernel_file_request {
00303     uint64_t id[4];
00304     uint64_t revision;
00305     LIMINE_PTR(struct limine_kernel_file_response *) response;
00306 };
00307
00308 /* Module */
00309
00310 #define LIMINE_MODULE_REQUEST { LIMINE_COMMON_MAGIC, 0x3e7e279702be32af, 0xcacfc4f3bd1280cee }
00311
00312 struct limine_module_response {
00313     uint64_t revision;
00314     uint64_t module_count;
00315     LIMINE_PTR(struct limine_file **) modules;
00316 };
00317
00318 struct limine_module_request {
00319     uint64_t id[4];
00320     uint64_t revision;
00321     LIMINE_PTR(struct limine_module_response *) response;
00322 };
00323
00324 /* RSDP */
00325
00326 #define LIMINE_RSDP_REQUEST { LIMINE_COMMON_MAGIC, 0xc5e77b6b397e7b43, 0x27637845accdf3c }
00327
00328 struct limine_rsdp_response {
00329     uint64_t revision;
00330     LIMINE_PTR(void *) address;
00331 };
00332
00333 struct limine_rsdp_request {
00334     uint64_t id[4];
00335     uint64_t revision;
00336     LIMINE_PTR(struct limine_rsdp_response *) response;
00337 };
00338
00339 /* SMBIOS */
00340
00341 #define LIMINE_SMBIOS_REQUEST { LIMINE_COMMON_MAGIC, 0x9e9046f11e095391, 0xaa4a520fefbde5ee }
00342
00343 struct limine_smbios_response {
00344     uint64_t revision;
00345     LIMINE_PTR(void *) entry_32;
```

```

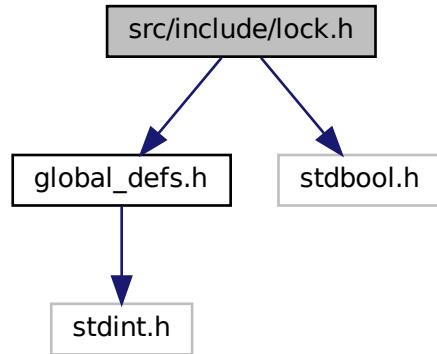
00346     LIMINE_PTR(void *) entry_64;
00347 };
00348
00349 struct limine_smbios_request {
00350     uint64_t id[4];
00351     uint64_t revision;
00352     LIMINE_PTR(struct limine_smbios_response *) response;
00353 };
00354
00355 /* EFI system table */
00356
00357 #define LIMINE_EFI_SYSTEM_TABLE_REQUEST { LIMINE_COMMON_MAGIC, 0x5ceba5163eaaf6d6, 0x0a6981610cf65fcc
00358 }
00359
00360 struct limine_efi_system_table_response {
00361     uint64_t revision;
00362     LIMINE_PTR(void *) address;
00363 }
00364
00365 struct limine_efi_system_table_request {
00366     uint64_t id[4];
00367     uint64_t revision;
00368     LIMINE_PTR(struct limine_efi_system_table_response *) response;
00369 };
00370
00371 /* Boot time */
00372
00373 #define LIMINE_BOOT_TIME_REQUEST { LIMINE_COMMON_MAGIC, 0x502746e184c088aa, 0xfbcd5ec83e6327893 }
00374
00375 struct limine_boot_time_response {
00376     uint64_t revision;
00377     int64_t boot_time;
00378 }
00379
00380 struct limine_boot_time_request {
00381     uint64_t id[4];
00382     uint64_t revision;
00383     LIMINE_PTR(struct limine_boot_time_response *) response;
00384 };
00385
00386 /* Kernel address */
00387
00388 #define LIMINE_KERNEL_ADDRESS_REQUEST { LIMINE_COMMON_MAGIC, 0x71ba76863cc55f63, 0xb2644a48c516a487 }
00389
00390 struct limine_kernel_address_response {
00391     uint64_t revision;
00392     uint64_t physical_base;
00393     uint64_t virtual_base;
00394 };
00395
00396 struct limine_kernel_address_request {
00397     uint64_t id[4];
00398     uint64_t revision;
00399     LIMINE_PTR(struct limine_kernel_address_response *) response;
00400 };
00401
00402 /* Device Tree Blob */
00403
00404 #define LIMINE_DTB_REQUEST { LIMINE_COMMON_MAGIC, 0xb40ddb48fb54bac7, 0x545081493f81ffb7 }
00405
00406 struct limine_dtb_response {
00407     uint64_t revision;
00408     LIMINE_PTR(void *) dtb_ptr;
00409 };
00410
00411 struct limine_dtb_request {
00412     uint64_t id[4];
00413     uint64_t revision;
00414     LIMINE_PTR(struct limine_dtb_response *) response;
00415
00416 #ifdef __cplusplus
00417 }
00418 #endif
00419
00420 #endif

```

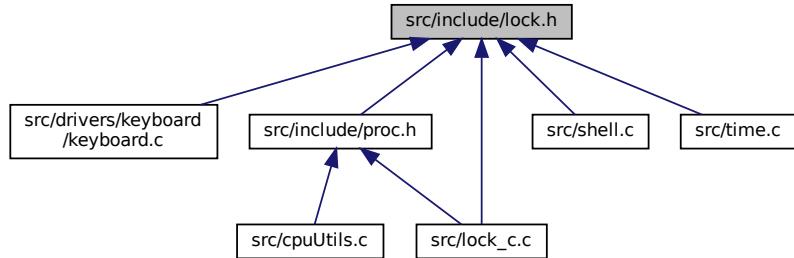
## 4.49 src/include/lock.h File Reference

```
#include "global_defs.h"
#include <stdbool.h>
```

Include dependency graph for lock.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `smartlock`

## Macros

- `#define _LOCK_C_H`
- `#define SPINLOCK_INIT 0`
- `#define SMARTLOCK_INIT`

## Typedefs

- `typedef int spinlock_t`

## Functions

- int `atomic_lock ()`
- int `atomic_unlock ()`
- void `smartlock_acquire (struct smartlock *smartlock)`
- void `smartlock_release (struct smartlock *smartlock)`
- static bool `spinlock_test_and_acq (spinlock_t *lock)`
- static void `spinlock_acquire (spinlock_t *lock)`
- static void `spinlock_release (spinlock_t *lock)`

## 4.49.1 Data Structure Documentation

### 4.49.1.1 struct smartlock

Definition at line 17 of file `lock.h`.

#### Data Fields

<code>size_t</code>	<code>refcount</code>	
<code>struct thread *</code>	<code>thread</code>	

## 4.49.2 Macro Definition Documentation

### 4.49.2.1 \_LOCK\_C\_H

```
#define _LOCK_C_H
```

Definition at line 4 of file `lock.h`.

### 4.49.2.2 SMARTLOCK\_INIT

```
#define SMARTLOCK_INIT
```

#### Value:

```
{ 0, NULL }
```

Definition at line 23 of file `lock.h`.

#### 4.49.2.3 SPINLOCK\_INIT

```
#define SPINLOCK_INIT 0
```

Definition at line 15 of file [lock.h](#).

### 4.49.3 Typedef Documentation

#### 4.49.3.1 spinlock\_t

```
typedef int spinlock_t
```

Definition at line 13 of file [lock.h](#).

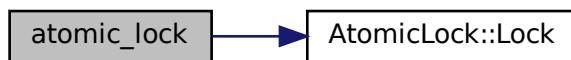
### 4.49.4 Function Documentation

#### 4.49.4.1 atomic\_lock()

```
int atomic_lock ( )
```

Definition at line 5 of file [lock\\_atm\\_c.cpp](#).

Here is the call graph for this function:



#### 4.49.4.2 atomic\_unlock()

```
int atomic_unlock ( )
```

Definition at line 13 of file [lock\\_atm\\_c.cpp](#).

Here is the call graph for this function:



#### 4.49.4.3 smartlock\_acquire()

```
void smartlock_acquire (
    struct smartlock * smartlock )
```

#### 4.49.4.4 smartlock\_release()

```
void smartlock_release (
    struct smartlock * smartlock )
```

#### 4.49.4.5 spinlock\_acquire()

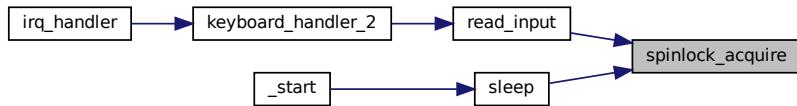
```
static void spinlock_acquire (
    spinlock_t * lock ) [inline], [static]
```

Definition at line 36 of file [lock.h](#).

Here is the call graph for this function:



Here is the caller graph for this function:

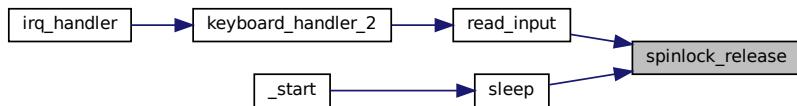


#### 4.49.4.6 spinlock\_release()

```
static void spinlock_release (
    spinlock_t * lock ) [inline], [static]
```

Definition at line 50 of file [lock.h](#).

Here is the caller graph for this function:

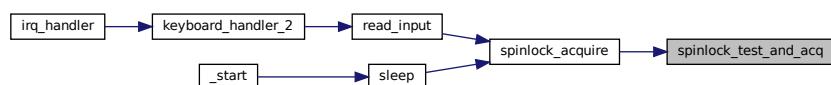


#### 4.49.4.7 spinlock\_test\_and\_acq()

```
static bool spinlock_test_and_acq (
    spinlock_t * lock ) [inline], [static]
```

Definition at line 31 of file [lock.h](#).

Here is the caller graph for this function:



## 4.50 lock.h

```

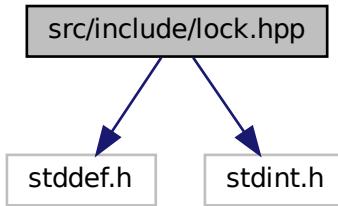
00001 #pragma once
00002
00003 #ifndef _LOCK_C_H
00004 #define _LOCK_C_H
00005
00006 #include "global_defs.h"
00007 #include <stdbool.h>
00008
00009 extern int atomic_lock();
00010
00011 extern int atomic_unlock();
00012
00013 typedef int spinlock_t;
00014
00015 #define SPINLOCK_INIT 0
00016
00017 struct smartlock
00018 {
00019     size_t refcount;
00020     struct thread *thread;
00021 };
00022
00023 #define SMARTLOCK_INIT \
00024     { \
00025         0, NULL \
00026     }
00027
00028 void smartlock_acquire(struct smartlock *smartlock);
00029 void smartlock_release(struct smartlock *smartlock);
00030
00031 static inline bool spinlock_test_and_acq(spinlock_t *lock)
00032 {
00033     return CAS(lock, 0, 1);
00034 }
00035
00036 static inline void spinlock_acquire(spinlock_t *lock)
00037 {
00038     for (;;)
00039     {
00040         if (spinlock_test_and_acq(lock))
00041         {
00042             break;
00043         }
00044 #if defined(__x86_64__)
00045     asm volatile("pause");
00046 #endif
00047     }
00048 }
00049
00050 static inline void spinlock_release(spinlock_t *lock)
00051 {
00052     CAS(lock, 1, 0);
00053 }
00054
00055 #endif

```

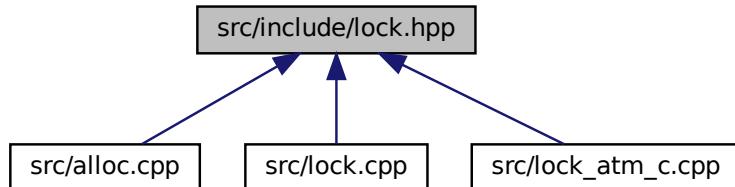
## 4.51 src/include/lock.hpp File Reference

```
#include <stddef.h>
#include <stdint.h>
```

Include dependency graph for lock.hpp:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [AtomicLock](#)
- class [ScopedLock](#)

## 4.52 lock.hpp

```
00001 #pragma once
00002
00003 #include <stddef.h>
00004 #include <stdint.h>
00005
00006 class AtomicLock
00007 {
00008     private:
00009         volatile uint32_t locked;
00010
00011     public:
00012         AtomicLock();
00013
00014     bool IsLocked() const;
00015
00016     void Lock();
00017     void Unlock();
00018
00019     void ForceLock();
00020 };
00021
```

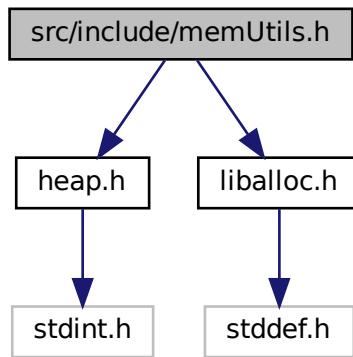
```

00022 class ScopedLock
00023 {
00024     private:
00025         AtomicLock &lock;
00026
00027     public:
00028     ScopedLock(AtomicLock &value);
00029     ~ScopedLock();
00030 };

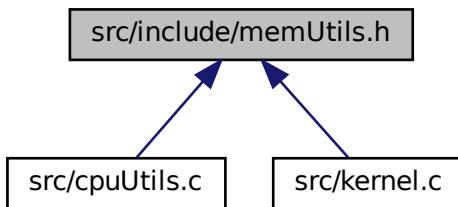
```

## 4.53 src/include/memUtils.h File Reference

```
#include "heap.h"
#include "liballoc.h"
Include dependency graph for memUtils.h:
```



This graph shows which files directly or indirectly include this file:



### Macros

- #define **MEMUTILS\_H**
- #define **SIZE** 0x500000
- #define **BSIZE** 16
- #define **ALLOC**(TYPE) ([malloc\(sizeof\(TYPE\)\)](#))

## Variables

- KHEAPBM kheap

### 4.53.1 Macro Definition Documentation

#### 4.53.1.1 ALLOC

```
#define ALLOC( TYPE ) (malloc(sizeof(TYPE)))
```

Definition at line 13 of file [memUtils.h](#).

#### 4.53.1.2 BSIZE

```
#define BSIZE 16
```

Definition at line 11 of file [memUtils.h](#).

#### 4.53.1.3 MEMUTILS\_H

```
#define MEMUTILS_H
```

Definition at line 3 of file [memUtils.h](#).

#### 4.53.1.4 SIZE

```
#define SIZE 0x500000
```

Definition at line 10 of file [memUtils.h](#).

### 4.53.2 Variable Documentation

### 4.53.2.1 kheap

```
KHEAPBM kheap [extern]
```

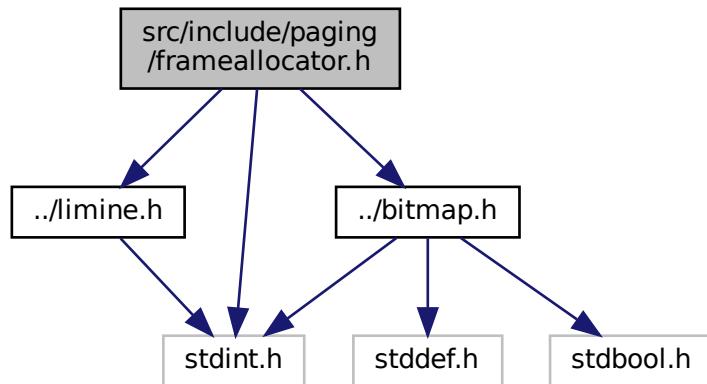
Definition at line 39 of file [kernel.c](#).

## 4.54 memUtils.h

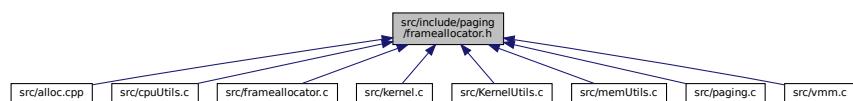
```
00001 #pragma once
00002 #ifndef MEMUTILS_H
00003 #define MEMUTILS_H
00004
00005 #include "heap.h"
00006 #include "liballoc.h"
00007
00008 extern KHEAPBM kheap;
00009
00010 #define SIZE 0x500000
00011 #define BSIZE 16
00012
00013 #define ALLOC(TYPE) (malloc(sizeof(TYPE)))
00014
00015 #endif
```

## 4.55 src/include/paging/frameallocator.h File Reference

```
#include "../limine.h"
#include <stdint.h>
#include "../bitmap.h"
Include dependency graph for frameallocator.h:
```



This graph shows which files directly or indirectly include this file:



## Macros

- `#define FRAME_EXPORT`
- `#define PAGE_SIZE 4096`

## Functions

- `void read_memory_map ()`
- `void frame_free (void *address)`
- `FRAME_EXPORT void frame_free_multiple (void *address, uint64_t pageCount)`
- `void frame_lock (void *address)`
- `void frame_lock_multiple (void *address, uint64_t pageCount)`
- `void * frame_request ()`
- `FRAME_EXPORT void * frame_request_multiple (uint32_t count)`
- `uint64_t free_ram ()`
- `uint64_t used_ram ()`
- `uint64_t reserved_ram ()`
- `void print_frame_bitmap ()`

### 4.55.1 Macro Definition Documentation

#### 4.55.1.1 FRAME\_EXPORT

```
#define FRAME_EXPORT
```

Definition at line 9 of file [frameallocator.h](#).

#### 4.55.1.2 PAGE\_SIZE

```
#define PAGE_SIZE 4096
```

Definition at line 12 of file [frameallocator.h](#).

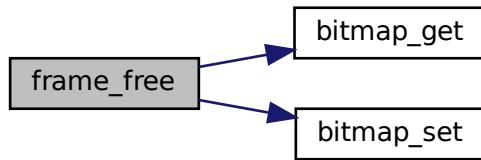
### 4.55.2 Function Documentation

#### 4.55.2.1 frame\_free()

```
void frame_free (
    void * address )
```

Definition at line 207 of file [frameallocator.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

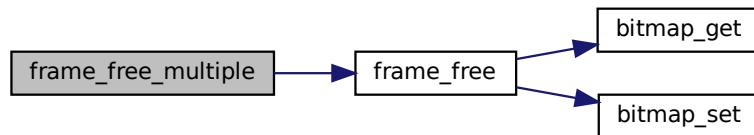


#### 4.55.2.2 frame\_free\_multiple()

```
FRAME_EXPORT void frame_free_multiple (
    void * address,
    uint64_t pageCount )
```

Definition at line 222 of file [frameallocator.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

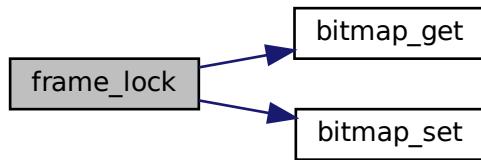


#### 4.55.2.3 frame\_lock()

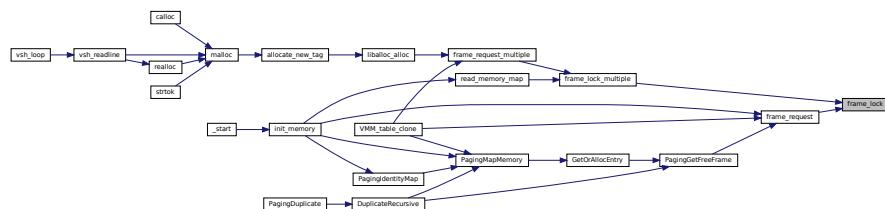
```
void frame_lock (
    void * address )
```

Definition at line 230 of file [frameallocator.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

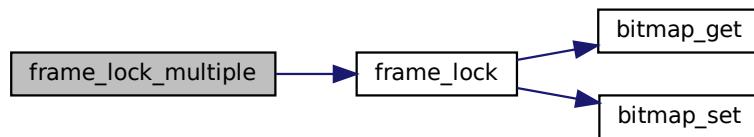


#### 4.55.2.4 frame\_lock\_multiple()

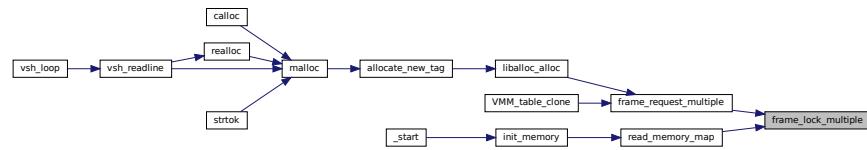
```
void frame_lock_multiple (
    void * address,
    uint64_t pageCount )
```

Definition at line 245 of file [frameallocator.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

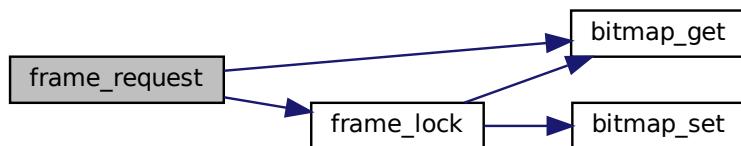


#### 4.55.2.5 frame\_request()

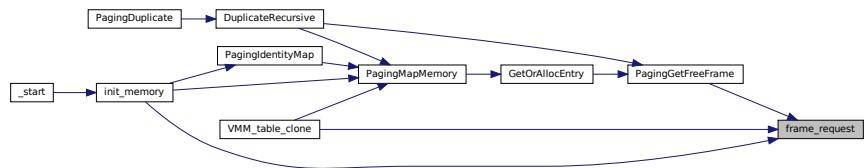
```
void* frame_request ( )
```

Definition at line 146 of file [frameallocator.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

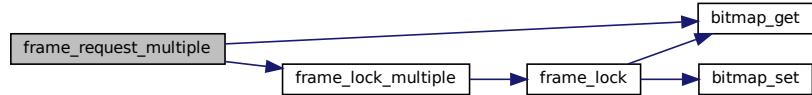


#### 4.55.2.6 frame\_request\_multiple()

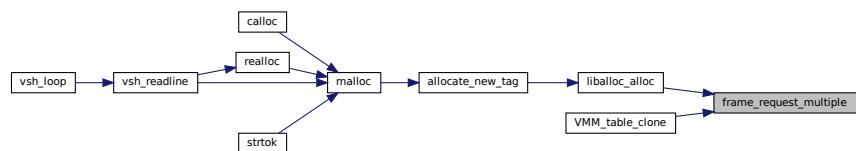
```
FRAME_EXPORT void* frame_request_multiple (
    uint32_t count )
```

Definition at line 179 of file [frameallocator.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

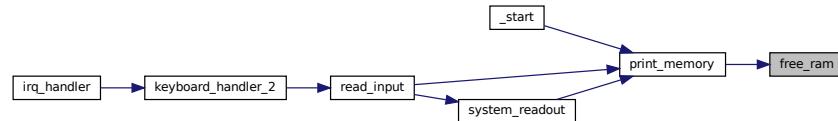


#### 4.55.2.7 free\_ram()

```
uint64_t free_ram ( )
```

Definition at line 253 of file [frameallocator.c](#).

Here is the caller graph for this function:

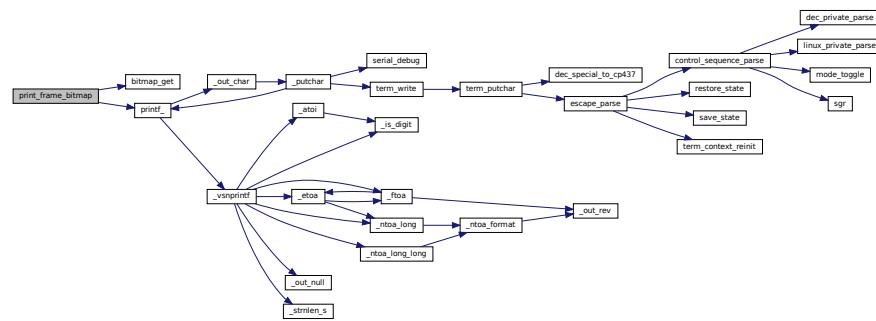


#### 4.55.2.8 print\_frame\_bitmap()

```
void print_frame_bitmap ( )
```

Definition at line 161 of file [frameallocator.c](#).

Here is the call graph for this function:

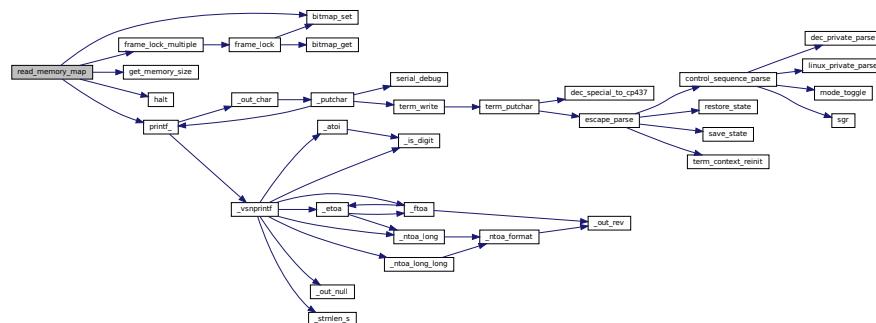


#### 4.55.2.9 read\_memory\_map()

```
void read_memory_map ( )
```

Definition at line 60 of file [frameallocator.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

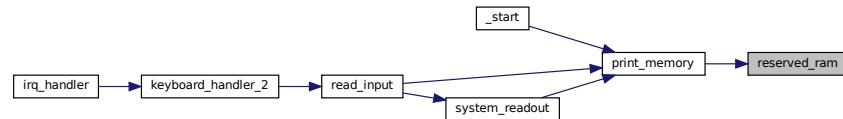


#### 4.55.2.10 reserved\_ram()

```
uint64_t reserved_ram ( )
```

Definition at line 263 of file [frameallocator.c](#).

Here is the caller graph for this function:

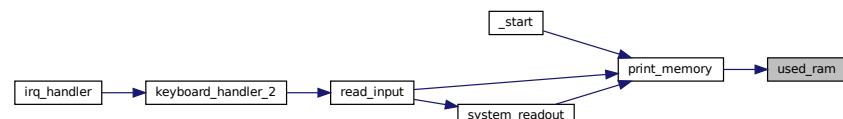


#### 4.55.2.11 used\_ram()

```
uint64_t used_ram ( )
```

Definition at line 258 of file [frameallocator.c](#).

Here is the caller graph for this function:



## 4.56 frameallocator.h

```

00001 #pragma once
00002 #include "../limine.h"
00003 #include <stdint.h>
00004 #include "../bitmap.h"
00005
00006 #ifdef __cplusplus
00007 #define FRAME_EXPORT extern "C"
00008 #else
00009 #define FRAME_EXPORT
00010 #endif
00011
00012 #define PAGE_SIZE 4096
00013
00014 void read_memory_map();
00015
00016 void frame_free(void *address);
00017 FRAME_EXPORT void frame_free_multiple(void *address, uint64_t pageCount);
00018 void frame_lock(void *address);
00019 void frame_lock_multiple(void *address, uint64_t pageCount);
00020 void *frame_request();
00021 FRAME_EXPORT void *frame_request_multiple(uint32_t count);
00022 uint64_t free_ram();
00023 uint64_t used_ram();
00024 uint64_t reserved_ram();
00025 void print_frame_bitmap();
00026 // file contains the PMM functions

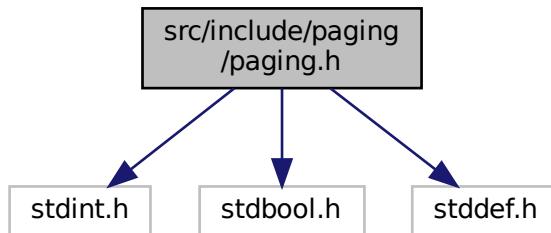
```

## 4.57 src/include/paging/paging.h File Reference

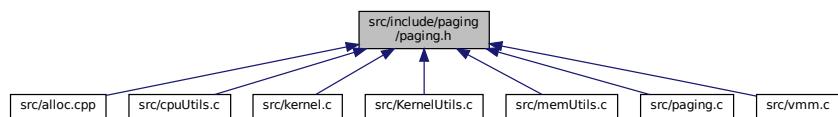
```

#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>
Include dependency graph for paging.h:

```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `PageTableOffset`

## Macros

- `#define PAGING_EXPORT`
- `#define HIGHER_HALF_MEMORY_OFFSET 0xFFFF800000000000`
- `#define HIGHER_HALF_KERNEL_MEMORY_OFFSET 0xFFFFFFFF80000000`
- `#define PAGE_FLAG_MASK (0xFFF | (1ull << 63))`
- `#define PAGE_ADDRESS_MASK (~(PAGE_FLAG_MASK))`

## Enumerations

- enum `PagingFlag` {
   
`PAGING_FLAG_PRESENT = (1ull << 0)` , `PAGING_FLAG_WRITABLE = (1ull << 1)` , `PAGING_FLAG_USER_ACCESSIBLE = (1ull << 2)` , `PAGING_FLAG_PAT0 = (1ull << 3)` ,
   
`PAGING_FLAG_PAT1 = (1ull << 4)` , `PAGING_FLAG_PAT2 = (1ull << 7)` , `PAGING_FLAG_WRITE_COMBINE = PAGING_FLAG_PAT0 | PAGING_FLAG_PAT1` , `PAGING_FLAG_LARGER_PAGES = (1ull << 7)` ,
 `PAGING_FLAG_NO_EXECUTE = (1ull << 63)` }

## Functions

- `uint64_t TranslateToHighHalfMemoryAddress (uint64_t physicalAddress)`
- `uint64_t TranslateToPhysicalMemoryAddress (uint64_t virtualAddress)`
- `uint64_t TranslateToKernelPhysicalMemoryAddress (uint64_t virtualAddress)`
- `uint64_t TranslateToKernelMemoryAddress (uint64_t virtualAddress)`
- `bool IsHigherHalf (uint64_t physicalAddress)`
- `bool IsKernelHigherHalf (uint64_t physicalAddress)`
- struct `__attribute__ ((aligned(0x1000))) PageTable`
- `PAGING_EXPORT void PagingMapMemory (struct PageTable *p4, void *virtualMemory, void *physicalMemory, uint64_t flags)`
- `PAGING_EXPORT void PagingUnmapMemory (struct PageTable *p4, void *virtualMemory)`
- `PAGING_EXPORT void PagingIdentityMap (struct PageTable *p4, void *virtualMemory, uint64_t flags)`
- `PAGING_EXPORT void * PagingPhysicalMemory (struct PageTable *p4, void *virtualMemory)`
- `PAGING_EXPORT void PagingDuplicate (struct PageTable *p4, struct PageTable *newTable)`
- `PAGING_EXPORT void PagingSetActivePageTable (struct PageTable *p4)`
- `PAGING_EXPORT uint64_t PagingGetFreeFrame ()`

### 4.57.1 Data Structure Documentation

#### 4.57.1.1 struct `PageTableOffset`

Definition at line 62 of file `paging.h`.

##### Data Fields

<code>uint64_t</code>	<code>p4Offset</code>	
<code>uint64_t</code>	<code>pdOffset</code>	
<code>uint64_t</code>	<code>pdpOffset</code>	
<code>uint64_t</code>	<code>ptOffset</code>	

## 4.57.2 Macro Definition Documentation

### 4.57.2.1 HIGHER\_HALF\_KERNEL\_MEMORY\_OFFSET

```
#define HIGHER_HALF_KERNEL_MEMORY_OFFSET 0xFFFFFFFF80000000
```

Definition at line 14 of file [paging.h](#).

### 4.57.2.2 HIGHER\_HALF\_MEMORY\_OFFSET

```
#define HIGHER_HALF_MEMORY_OFFSET 0xFFFF800000000000
```

Definition at line 13 of file [paging.h](#).

### 4.57.2.3 PAGE\_ADDRESS\_MASK

```
#define PAGE_ADDRESS_MASK (~(PAGE_FLAG_MASK))
```

Definition at line 17 of file [paging.h](#).

### 4.57.2.4 PAGE\_FLAG\_MASK

```
#define PAGE_FLAG_MASK (0xFFF | (ull << 63))
```

Definition at line 16 of file [paging.h](#).

### 4.57.2.5 PAGING\_EXPORT

```
#define PAGING_EXPORT
```

Definition at line 10 of file [paging.h](#).

## 4.57.3 Enumeration Type Documentation

### 4.57.3.1 PagingFlag

```
enum PagingFlag
```

Enumerator

PAGING_FLAG_PRESENT	
PAGING_FLAG_WRITABLE	
PAGING_FLAG_USER_ACCESSIBLE	
PAGING_FLAG_PAT0	
PAGING_FLAG_PAT1	
PAGING_FLAG_PAT2	
PAGING_FLAG_WRITE_COMBINE	
PAGING_FLAG_LARGER_PAGES	
PAGING_FLAG_NO_EXECUTE	

Definition at line 49 of file [paging.h](#).

#### 4.57.4 Function Documentation

##### 4.57.4.1 \_\_attribute\_\_()

```
struct __attribute__ (
    aligned(0x1000))
```

Definition at line 44 of file [paging.h](#).

##### 4.57.4.2 IsHigherHalf()

```
bool IsHigherHalf (
    uint64_t physicalAddress) [inline]
```

Definition at line 39 of file [paging.h](#).

##### 4.57.4.3 IsKernelHigherHalf()

```
bool IsKernelHigherHalf (
    uint64_t physicalAddress) [inline]
```

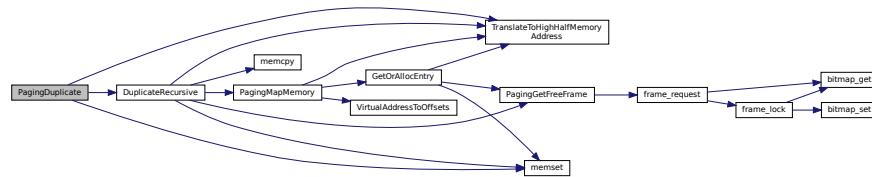
Definition at line 44 of file [paging.h](#).

#### 4.57.4.4 PagingDuplicate()

```
PAGING_EXPORT void PagingDuplicate (
    struct PageTable * p4,
    struct PageTable * newTable )
```

Definition at line 194 of file [paging.c](#).

Here is the call graph for this function:

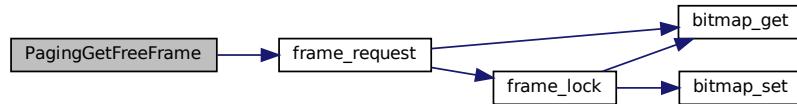


#### 4.57.4.5 PagingGetFreeFrame()

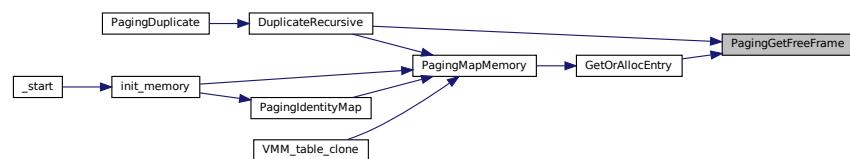
```
PAGING_EXPORT uint64_t PagingGetFreeFrame ( )
```

Definition at line 4 of file [memUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

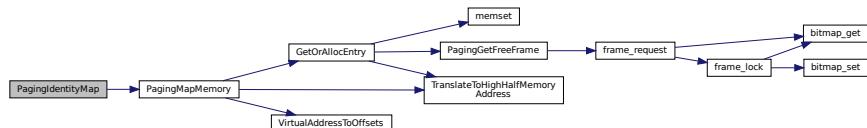


#### 4.57.4.6 PagingIdentityMap()

```
PAGING_EXPORT void PagingIdentityMap (
    struct PageTable * p4,
    void * virtualMemory,
    uint64_t flags )
```

Definition at line 115 of file [paging.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

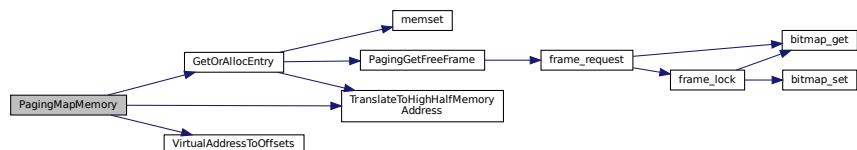


#### 4.57.4.7 PagingMapMemory()

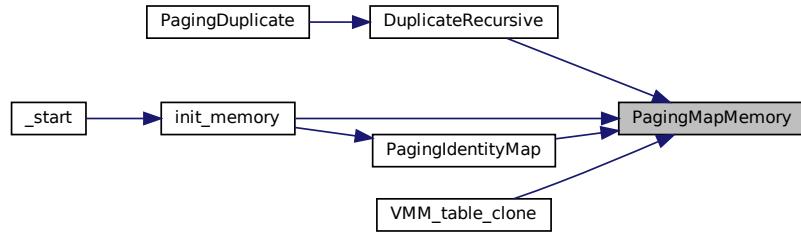
```
PAGING_EXPORT void PagingMapMemory (
    struct PageTable * p4,
    void * virtualMemory,
    void * physicalMemory,
    uint64_t flags )
```

Definition at line 120 of file [paging.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

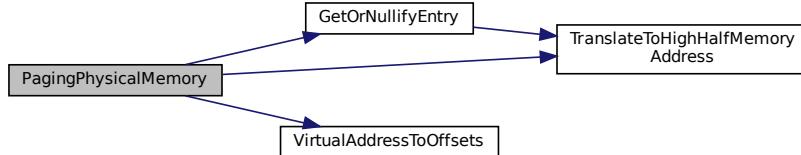


#### 4.57.4.8 PagingPhysicalMemory()

```
PAGING_EXPORT void* PagingPhysicalMemory (
    struct PageTable * p4,
    void * virtualMemory )
```

Definition at line 135 of file [paging.c](#).

Here is the call graph for this function:



#### 4.57.4.9 PagingSetActivePageTable()

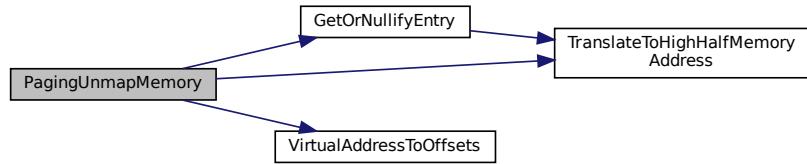
```
PAGING_EXPORT void PagingSetActivePageTable (
    struct PageTable * p4 )
```

#### 4.57.4.10 PagingUnmapMemory()

```
PAGING_EXPORT void PagingUnmapMemory (
    struct PageTable * p4,
    void * virtualMemory )
```

Definition at line 165 of file [paging.c](#).

Here is the call graph for this function:

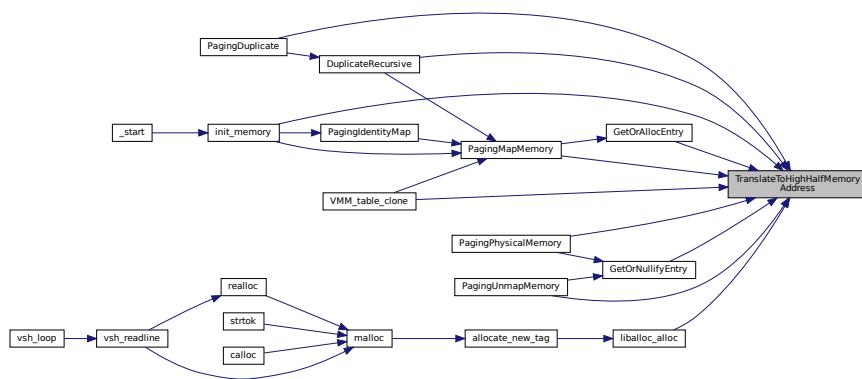


#### 4.57.4.11 TranslateToHighHalfMemoryAddress()

```
uint64_t TranslateToHighHalfMemoryAddress (
    uint64_t physicalAddress ) [inline]
```

Definition at line 19 of file [paging.h](#).

Here is the caller graph for this function:



#### 4.57.4.12 TranslateToKernelMemoryAddress()

```
uint64_t TranslateToKernelMemoryAddress (
    uint64_t virtualAddress ) [inline]
```

Definition at line 34 of file [paging.h](#).

#### 4.57.4.13 TranslateToKernelPhysicalMemoryAddress()

```
uint64_t TranslateToKernelPhysicalMemoryAddress (
    uint64_t virtualAddress ) [inline]
```

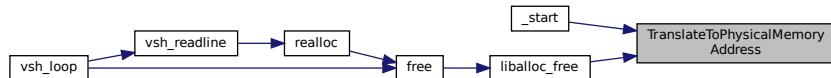
Definition at line 29 of file [paging.h](#).

#### 4.57.4.14 TranslateToPhysicalMemoryAddress()

```
uint64_t TranslateToPhysicalMemoryAddress (
    uint64_t virtualAddress ) [inline]
```

Definition at line 24 of file [paging.h](#).

Here is the caller graph for this function:



## 4.58 paging.h

```

00001 #ifndef _PAGING_H_
00002 #define _PAGING_H_
00003 #include <stdint.h>
00004 #include <stdbool.h>
00005 #include <stddef.h>
00006
00007 #ifdef __cplusplus
00008 #define PAGING_EXPORT extern "C"
00009 #else
00010 #define PAGING_EXPORT
00011 #endif
00012
00013 #define HIGHER_HALF_MEMORY_OFFSET 0xFFFF800000000000
00014 #define HIGHER_HALF_KERNEL_MEMORY_OFFSET 0xFFFFFFF80000000
00015
00016 #define PAGE_FLAG_MASK (0xFF | (ull << 63))
00017 #define PAGE_ADDRESS_MASK (~(PAGE_FLAG_MASK))
00018
00019 inline uint64_t TranslateToHighHalfMemoryAddress(uint64_t physicalAddress)
00020 {
00021     return physicalAddress + HIGHER_HALF_MEMORY_OFFSET;
00022 }
00023
00024 inline uint64_t TranslateToPhysicalMemoryAddress(uint64_t virtualAddress)
  
```

```

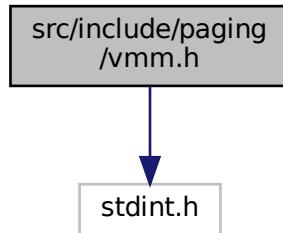
00025 {
00026     return virtualAddress - HIGHER_HALF_MEMORY_OFFSET;
00027 }
00028
00029 inline uint64_t TranslateToKernelPhysicalMemoryAddress(uint64_t virtualAddress)
00030 {
00031     return virtualAddress - HIGHER_HALF_KERNEL_MEMORY_OFFSET;
00032 }
00033
00034 inline uint64_t TranslateToKernelMemoryAddress(uint64_t virtualAddress)
00035 {
00036     return virtualAddress + HIGHER_HALF_KERNEL_MEMORY_OFFSET;
00037 }
00038
00039 inline bool IsHigherHalf(uint64_t physicalAddress)
00040 {
00041     return physicalAddress >= HIGHER_HALF_MEMORY_OFFSET;
00042 }
00043
00044 inline bool IsKernelHigherHalf(uint64_t physicalAddress)
00045 {
00046     return physicalAddress >= HIGHER_HALF_KERNEL_MEMORY_OFFSET;
00047 }
00048
00049 enum PagingFlag
00050 {
00051     PAGING_FLAG_PRESENT = (lull << 0),
00052     PAGING_FLAG_WRITABLE = (lull << 1),
00053     PAGING_FLAG_USER_ACCESSIBLE = (lull << 2),
00054     PAGING_FLAG_PAT0 = (lull << 3),
00055     PAGING_FLAG_PAT1 = (lull << 4),
00056     PAGING_FLAG_PAT2 = (lull << 7),
00057     PAGING_FLAG_WRITE_COMBINE = PAGING_FLAG_PATO | PAGING_FLAG_PAT1,
00058     PAGING_FLAG_LARGER_PAGES = (lull << 7),
00059     PAGING_FLAG_NO_EXECUTE = (lull << 63),
00060 };
00061
00062 struct PageTableOffset
00063 {
00064     uint64_t p4Offset;
00065     uint64_t pdpOffset;
00066     uint64_t pdOffset;
00067     uint64_t ptOffset;
00068 };
00069
00070 struct __attribute__((aligned(0x1000))) PageTable
00071 {
00072     uint64_t entries[512];
00073 };
00074
00075 PAGING_EXPORT void PagingMapMemory(struct PageTable *p4, void *virtualMemory, void *physicalMemory,
00076                                     uint64_t flags);
00076 PAGING_EXPORT void PagingUnmapMemory(struct PageTable *p4, void *virtualMemory);
00077 PAGING_EXPORT void PagingIdentityMap(struct PageTable *p4, void *virtualMemory, uint64_t flags);
00078 PAGING_EXPORT void *PagingPhysicalMemory(struct PageTable *p4, void *virtualMemory);
00079 PAGING_EXPORT void PagingDuplicate(struct PageTable *p4, struct PageTable *newTable);
00080 PAGING_EXPORT void PagingSetActivePageTable(struct PageTable *p4);
00081
00082 PAGING_EXPORT uint64_t PagingGetFreeFrame();
00083 #endif

```

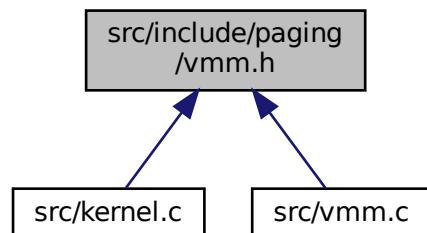
## 4.59 src/include/paging/vmm.h File Reference

```
#include <stdint.h>
```

Include dependency graph for vmm.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [dummy\\_proc](#)

## Functions

- void [VMM\\_table\\_clone \(\)](#)

### 4.59.1 Data Structure Documentation

#### 4.59.1.1 struct dummy\_proc

Definition at line [6](#) of file [vmm.h](#).

### Data Fields

uint64_t	data	
uint64_t	id	

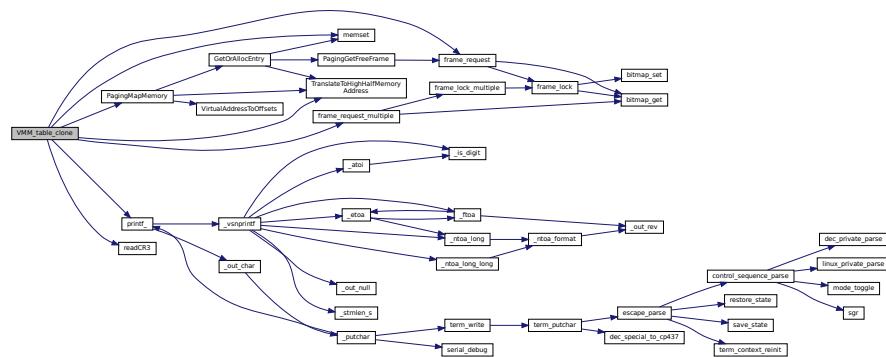
## 4.59.2 Function Documentation

### 4.59.2.1 VMM\_table\_clone()

```
void VMM_table_clone ( )
```

Definition at line 14 of file [vmm.c](#).

Here is the call graph for this function:



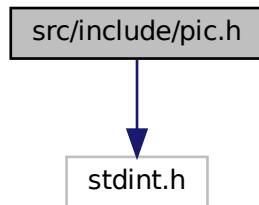
## 4.60 vmm.h

```
00001 #pragma once
00002 #include <stdint.h>
00003
00004 void VMM_table_clone();
00005
00006 struct dummy_proc
00007 {
00008
00009     uint64_t id;
00010     uint64_t data;
00011 };
```

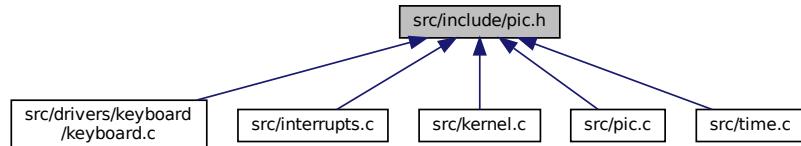
## 4.61 src/include/pic.h File Reference

```
#include "stdint.h"
```

Include dependency graph for pic.h:



This graph shows which files directly or indirectly include this file:



## Macros

- #define PIC\_MASTER 0x20
- #define PIC\_SLAVE 0xA0
- #define PIC\_MASTER\_COMMAND PIC\_MASTER
- #define PIC\_MASTER\_DATA (PIC\_MASTER + 1)
- #define PIC\_SLAVE\_COMMAND PIC\_SLAVE
- #define PIC\_SLAVE\_DATA (PIC\_SLAVE + 1)
- #define PIC\_EOI 0x20
- #define PIC\_ICW1\_ICW4 0x01
- #define PIC\_ICW1\_SINGLE 0x02
- #define PIC\_ICW1\_INTERVAL4 0x04
- #define PIC\_ICW1\_LEVEL 0x08
- #define PIC\_ICW1\_INIT 0x10
- #define PIC\_ICW4\_8086 0x01
- #define PIC\_ICW4\_AUTO 0x02
- #define PIC\_ICW4\_BUF\_SLAVE 0x08
- #define PIC\_ICW4\_BUF\_MASTER 0x0C

## Functions

- void [pic\\_enable](#) (void)
- void [pic\\_remap\\_offsets](#) (uint8\_t irq\_offset)
- void [pic\\_mask\\_irq](#) (uint8\_t irq)
- void [pic\\_unmask\\_irq](#) (uint8\_t irq)
- void [pic\\_send\\_eoi](#) (uint8\_t irq)

### 4.61.1 Macro Definition Documentation

#### 4.61.1.1 PIC\_EOI

```
#define PIC_EOI 0x20
```

Definition at line [10](#) of file [pic.h](#).

#### 4.61.1.2 PIC\_ICW1\_ICW4

```
#define PIC_ICW1_ICW4 0x01
```

Definition at line [12](#) of file [pic.h](#).

#### 4.61.1.3 PIC\_ICW1\_INIT

```
#define PIC_ICW1_INIT 0x10
```

Definition at line [16](#) of file [pic.h](#).

#### 4.61.1.4 PIC\_ICW1\_INTERVAL4

```
#define PIC_ICW1_INTERVAL4 0x04
```

Definition at line [14](#) of file [pic.h](#).

#### 4.61.1.5 PIC\_ICW1\_LEVEL

```
#define PIC_ICW1_LEVEL 0x08
```

Definition at line 15 of file [pic.h](#).

#### 4.61.1.6 PIC\_ICW1\_SINGLE

```
#define PIC_ICW1_SINGLE 0x02
```

Definition at line 13 of file [pic.h](#).

#### 4.61.1.7 PIC\_ICW4\_8086

```
#define PIC_ICW4_8086 0x01
```

Definition at line 18 of file [pic.h](#).

#### 4.61.1.8 PIC\_ICW4\_AUTO

```
#define PIC_ICW4_AUTO 0x02
```

Definition at line 19 of file [pic.h](#).

#### 4.61.1.9 PIC\_ICW4\_BUF\_MASTER

```
#define PIC_ICW4_BUF_MASTER 0x0C
```

Definition at line 21 of file [pic.h](#).

#### 4.61.1.10 PIC\_ICW4\_BUF\_SLAVE

```
#define PIC_ICW4_BUF_SLAVE 0x08
```

Definition at line 20 of file [pic.h](#).

#### 4.61.1.11 PIC\_MASTER

```
#define PIC_MASTER 0x20
```

Definition at line [4](#) of file [pic.h](#).

#### 4.61.1.12 PIC\_MASTER\_COMMAND

```
#define PIC_MASTER_COMMAND PIC_MASTER
```

Definition at line [6](#) of file [pic.h](#).

#### 4.61.1.13 PIC\_MASTER\_DATA

```
#define PIC_MASTER_DATA (PIC_MASTER + 1)
```

Definition at line [7](#) of file [pic.h](#).

#### 4.61.1.14 PIC\_SLAVE

```
#define PIC_SLAVE 0xA0
```

Definition at line [5](#) of file [pic.h](#).

#### 4.61.1.15 PIC\_SLAVE\_COMMAND

```
#define PIC_SLAVE_COMMAND PIC_SLAVE
```

Definition at line [8](#) of file [pic.h](#).

#### 4.61.1.16 PIC\_SLAVE\_DATA

```
#define PIC_SLAVE_DATA (PIC_SLAVE + 1)
```

Definition at line [9](#) of file [pic.h](#).

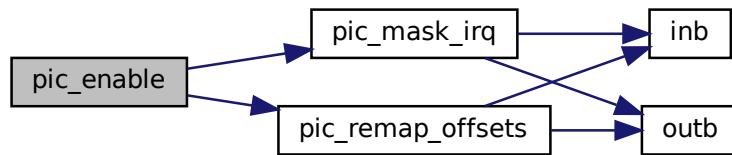
## 4.61.2 Function Documentation

### 4.61.2.1 pic\_enable()

```
void pic_enable (
    void  )
```

Definition at line 74 of file [pic.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

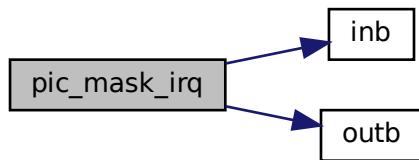


### 4.61.2.2 pic\_mask\_irq()

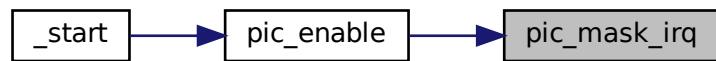
```
void pic_mask_irq (
    uint8_t irq )
```

Definition at line 4 of file [pic.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

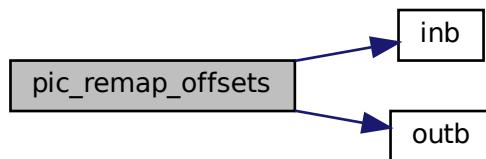


#### 4.61.2.3 `pic_remap_offsets()`

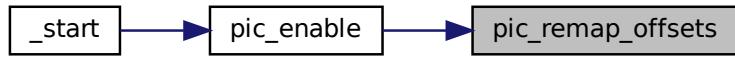
```
void pic_remap_offsets (
    uint8_t irq_offset )
```

Definition at line 44 of file [pic.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

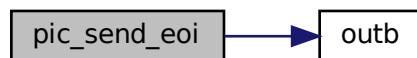


#### 4.61.2.4 pic\_send\_eoi()

```
void pic_send_eoi (
    uint8_t irq )
```

Definition at line 67 of file [pic.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

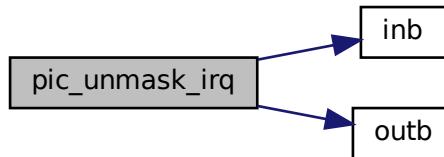


### 4.61.2.5 pic\_unmask\_irq()

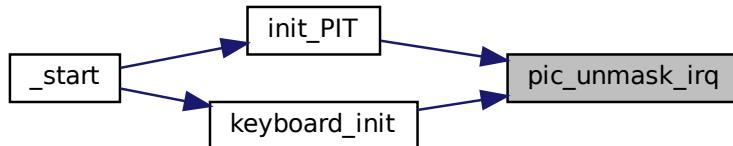
```
void pic_unmask_irq (
    uint8_t irq )
```

Definition at line 24 of file [pic.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

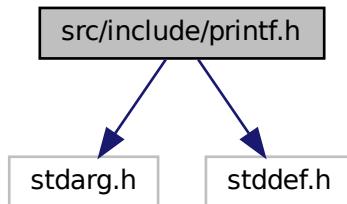


## 4.62 pic.h

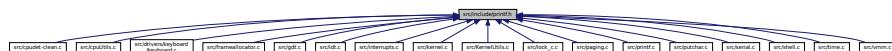
```
00001 #pragma once
00002 #include "stdint.h"
00003
00004 #define PIC_MASTER 0x20
00005 #define PIC_SLAVE 0xA0
00006 #define PIC_MASTER_COMMAND PIC_MASTER
00007 #define PIC_MASTER_DATA (PIC_MASTER + 1)
00008 #define PIC_SLAVE_COMMAND PIC_SLAVE
00009 #define PIC_SLAVE_DATA (PIC_SLAVE + 1)
00010 #define PIC_EOI 0x20
00011
00012 #define PIC_ICW1_ICW4 0x01
00013 #define PIC_ICW1_SINGLE 0x02
00014 #define PIC_ICW1_INTERVAL4 0x04
00015 #define PIC_ICW1_LEVEL 0x08
00016 #define PIC_ICW1_INIT 0x10
00017
00018 #define PIC_ICW4_8086 0x01
00019 #define PIC_ICW4_AUTO 0x02
00020 #define PIC_ICW4_BUF_SLAVE 0x08
00021 #define PIC_ICW4_BUF_MASTER 0x0C
00022
00023 void pic_enable(void);
00024 void pic_remap_offsets(uint8_t irq_offset);
00025 void pic_mask_irq(uint8_t irq);
00026 void pic_unmask_irq(uint8_t irq);
00027 void pic_send_eoi(uint8_t irq);
```

## 4.63 src/include/printf.h File Reference

```
#include <stdarg.h>
#include <stddef.h>
Include dependency graph for printf.h:
```



This graph shows which files directly or indirectly include this file:



### Macros

- #define \_PRINTF\_H\_
- #define printf printf\_
- #define sprintf sprintf\_
- #define snprintf snprintf\_
- #define vsnprintf vsnprintf\_
- #define vprintf vprintf\_

### Functions

- void putchar (char character)
- int printf\_ (const char \*format,...)
- int sprintf\_ (char \*buffer, const char \*format,...)
- int snprintf\_ (char \*buffer, size\_t count, const char \*format,...)
- int vsnprintf\_ (char \*buffer, size\_t count, const char \*format, va\_list va)
- int vprintf\_ (const char \*format, va\_list va)
- int fctprintf (void(\*out)(char character, void \*arg), void \*arg, const char \*format,...)

#### 4.63.1 Macro Definition Documentation

### 4.63.1.1 \_PRINTF\_H\_

```
#define _PRINTF_H_
```

Definition at line 34 of file [printf.h](#).

### 4.63.1.2 printf

```
#define printf printf_
```

Tiny printf implementation You have to implement *putchar* if you use *printf()* To avoid conflicts with the regular *printf()* API it is overridden by macro defines and internal underscore-appended functions like *printf()* are used

#### Parameters

<i>format</i>	A string that specifies the format of the output
---------------	--

#### Returns

The number of characters that are written into the array, not counting the terminating null character

Definition at line 59 of file [printf.h](#).

### 4.63.1.3 snprintf

```
#define snprintf snprintf_
```

Tiny snprintf/vsnprintf implementation

#### Parameters

<i>buffer</i>	A pointer to the buffer where to store the formatted string
<i>count</i>	The maximum number of characters to store in the buffer, including a terminating null character
<i>format</i>	A string that specifies the format of the output
<i>va</i>	A value identifying a variable arguments list

#### Returns

The number of characters that COULD have been written into the buffer, not counting the terminating null character. A value equal or larger than count indicates truncation. Only when the returned value is non-negative and less than count, the string has been completely written.

Definition at line 82 of file [printf.h](#).

#### 4.63.1.4 `sprintf`

```
#define sprintf sprintf_
```

Tiny sprintf implementation Due to security reasons (buffer overflow) YOU SHOULD CONSIDER USING (V)SNPRINTF INSTEAD!

##### Parameters

<i>buffer</i>	A pointer to the buffer where to store the formatted string. MUST be big enough to store the output!
<i>format</i>	A string that specifies the format of the output

##### Returns

The number of characters that are WRITTEN into the buffer, not counting the terminating null character

Definition at line [69](#) of file [printf.h](#).

#### 4.63.1.5 `vprintf`

```
#define vprintf vprintf_
```

Tiny vprintf implementation

##### Parameters

<i>format</i>	A string that specifies the format of the output
<i>va</i>	A value identifying a variable arguments list

##### Returns

The number of characters that are WRITTEN into the buffer, not counting the terminating null character

Definition at line [93](#) of file [printf.h](#).

#### 4.63.1.6 `vsnprintf`

```
#define vsnprintf vsnprintf_
```

Definition at line [83](#) of file [printf.h](#).

## 4.63.2 Function Documentation

### 4.63.2.1 \_putchar()

```
void _putchar (
    char character )
```

Output a character to a custom device like UART, used by the `printf()` function. This function is declared here only. You have to write your custom implementation somewhere.

#### Parameters

<code>character</code>	Character to output
------------------------	---------------------

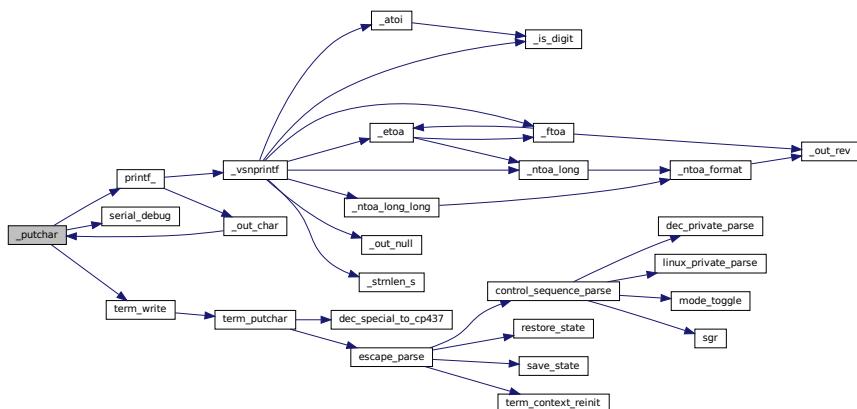
lets the printf function use the serial or terminal depending if the kernel has setup memory for the terminal or not.

#### Parameters

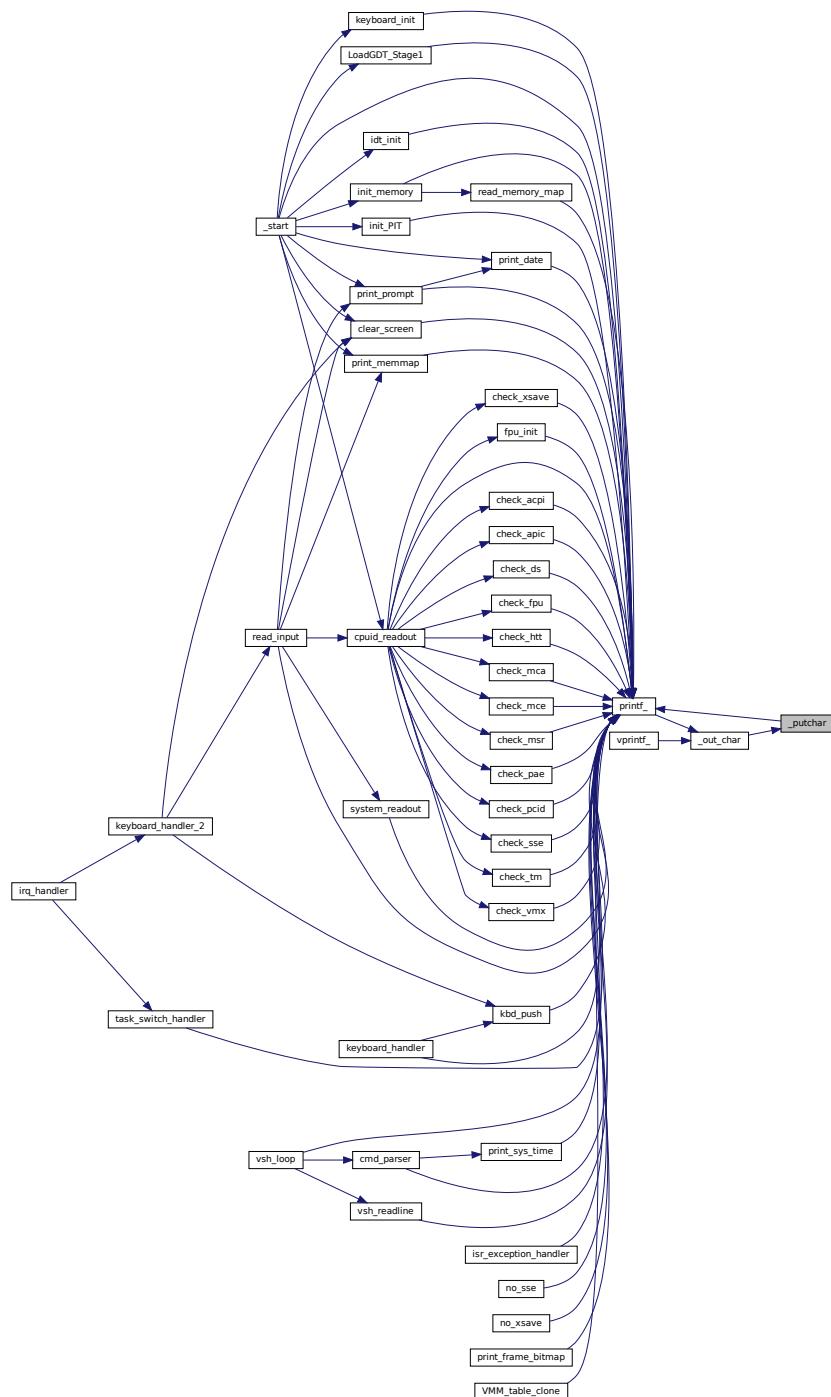
<code>character</code>	
------------------------	--

Definition at line 7 of file `putchar.c`.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.63.2.2 fctprintf()

```
int fctprintf (
    void(*)(char character, void *arg) out,
```

```
void * arg,
const char * format,
... )
```

printf with output function You may use this as dynamic alternative to [printf\(\)](#) with its fixed [\\_putchar\(\)](#) output

#### Parameters

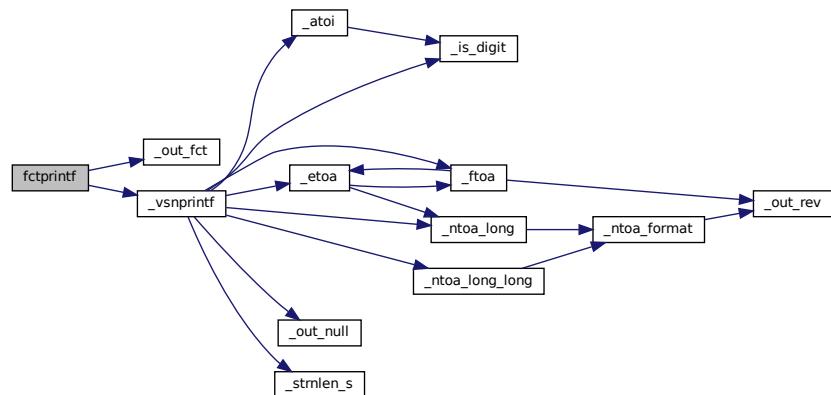
<i>out</i>	An output function which takes one character and an argument pointer
<i>arg</i>	An argument pointer for user data passed to output function
<i>format</i>	A string that specifies the format of the output

#### Returns

The number of characters that are sent to the output function, not counting the terminating null character

Definition at line 1036 of file [printf.c](#).

Here is the call graph for this function:

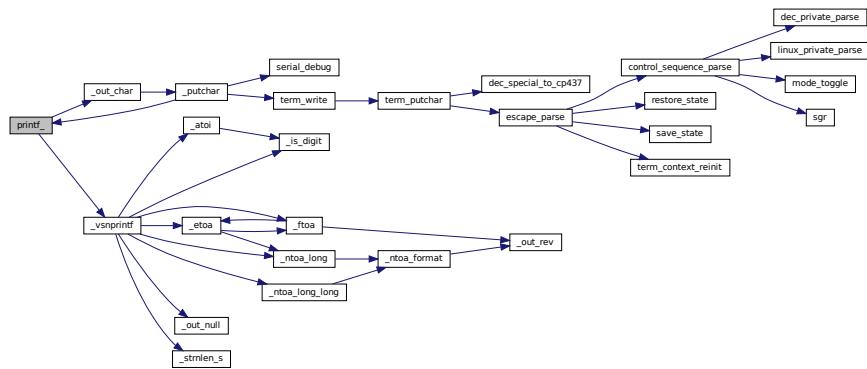


#### 4.63.2.3 printf\_()

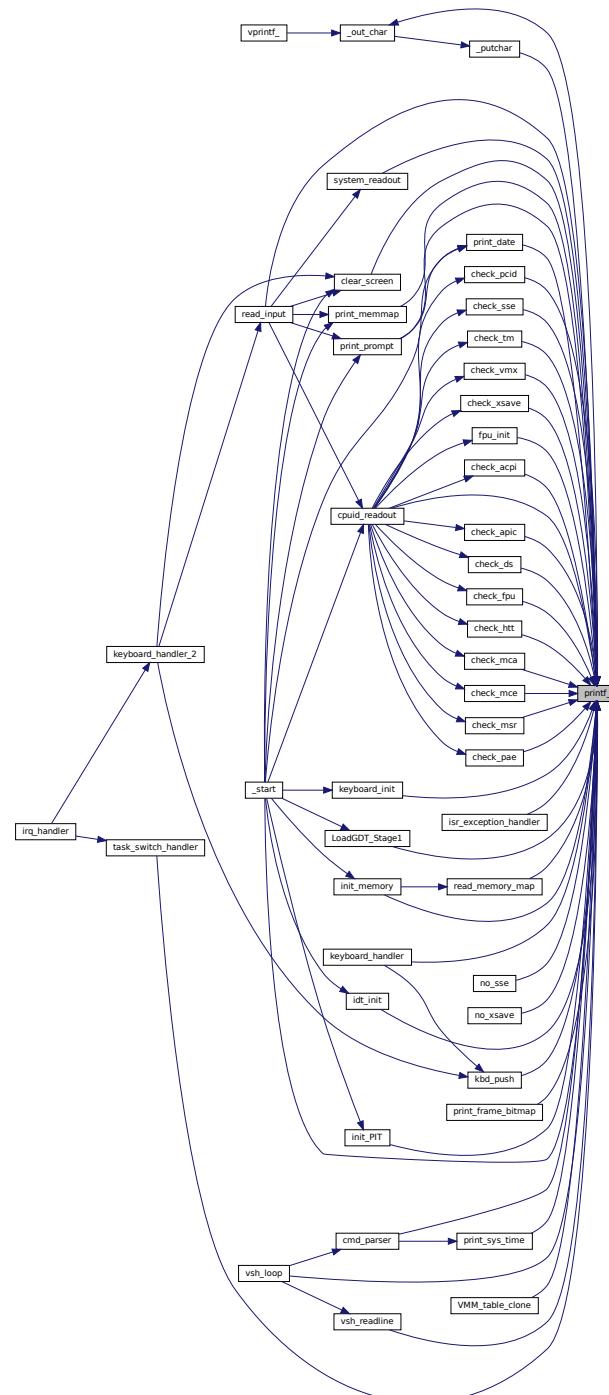
```
int printf_ (
    const char * format,
    ... )
```

Definition at line 997 of file [printf.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



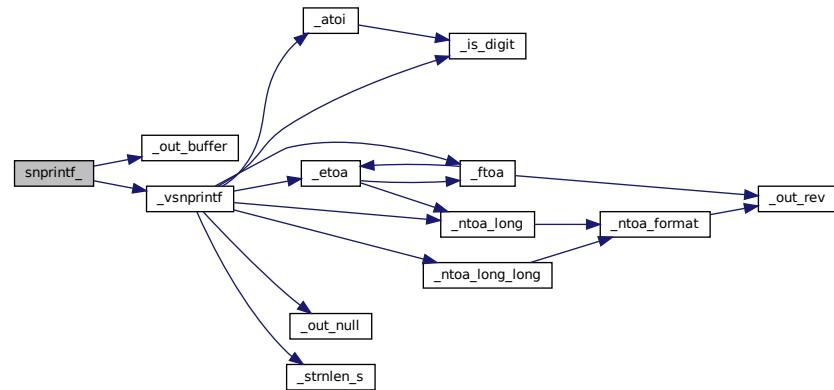
#### 4.63.2.4 snprintf\_()

```
int snprintf_(
    char * buffer,
```

```
size_t count,
const char * format,
... )
```

Definition at line 1016 of file [printf.c](#).

Here is the call graph for this function:

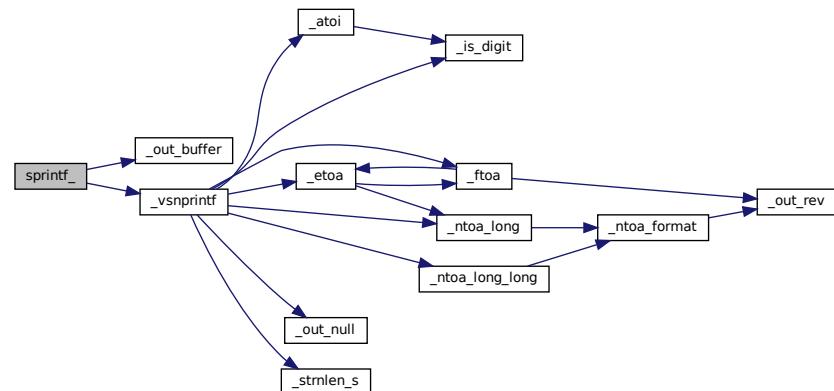


#### 4.63.2.5 sprintf\_()

```
int sprintf_ (
    char * buffer,
    const char * format,
    ... )
```

Definition at line 1007 of file [printf.c](#).

Here is the call graph for this function:

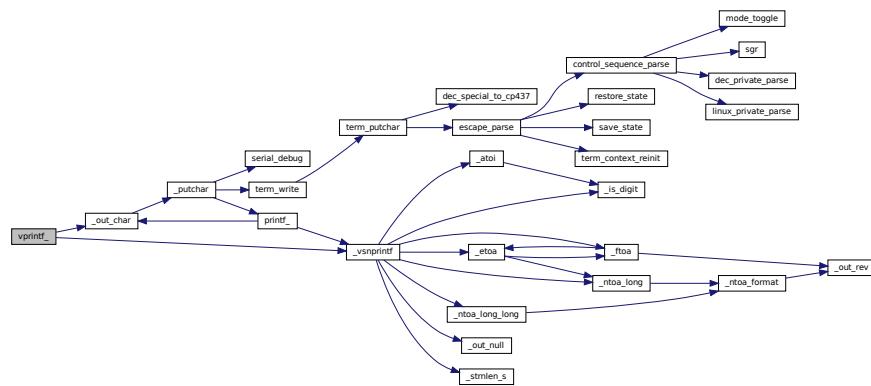


### 4.63.2.6 vprintf\_()

```
int vprintf_ (
    const char * format,
    va_list va )
```

Definition at line 1025 of file [printf.c](#).

Here is the call graph for this function:

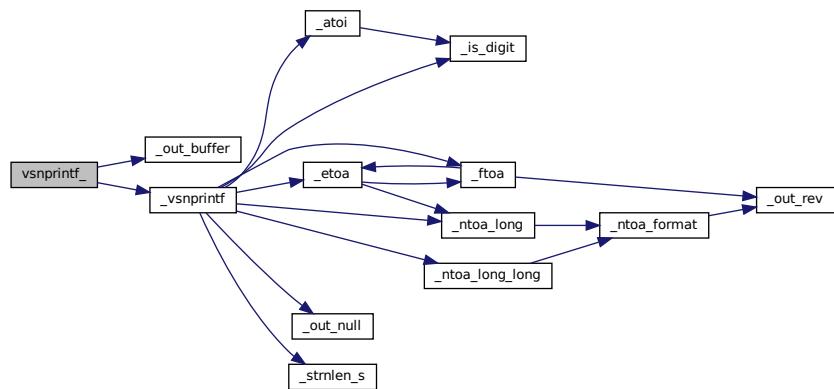


### 4.63.2.7 vsnprintf\_()

```
int vsnprintf_ (
    char * buffer,
    size_t count,
    const char * format,
    va_list va )
```

Definition at line 1031 of file [printf.c](#).

Here is the call graph for this function:



## 4.64 printf.h

```

00001 // \author (c) Marco Paland (info@paland.com)
00003 //           2014-2019, PALANDesign Hannover, Germany
00004 //
00005 // \license The MIT License (MIT)
00006 //
00007 // Permission is hereby granted, free of charge, to any person obtaining a copy
00008 // of this software and associated documentation files (the "Software"), to deal
00009 // in the Software without restriction, including without limitation the rights
00010 // to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00011 // copies of the Software, and to permit persons to whom the Software is
00012 // furnished to do so, subject to the following conditions:
00013 //
00014 // The above copyright notice and this permission notice shall be included in
00015 // all copies or substantial portions of the Software.
00016 //
00017 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00018 // IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00019 // FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00020 // AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00021 // LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00022 // OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
00023 // THE SOFTWARE.
00024 //
00025 // \brief Tiny printf, sprintf and snprintf implementation, optimized for speed on
00026 //        embedded systems with a very limited resources.
00027 //        Use this instead of bloated standard/newlib printf.
00028 //        These routines are thread safe and reentrant.
00029 //
00031
00032 #pragma once
00033 #ifndef __PRINTF_H_
00034 #define __PRINTF_H_
00035
00036 #include <stdarg.h>
00037 #include <stddef.h>
00038
00039 #ifdef __cplusplus
00040 extern "C"
00041 {
00042 #endif
00043
00044     void _putchar(char character);
00050
00059 #define printf printf_
00060     int printf_(const char *format, ...);
00061
00069 #define sprintf sprintf_
00070     int sprintf_(char *buffer, const char *format, ...);
00071
00082 #define snprintf snprintf_
00083 #define vsnprintf vsnprintf_
00084     int snprintf_(char *buffer, size_t count, const char *format, ...);
00085     int vsnprintf_(char *buffer, size_t count, const char *format, va_list va);
00086
00093 #define vprintf vprintf_
00094     int vprintf_(const char *format, va_list va);
00095
00104     int fctprintf(void (*out)(char character, void *arg), void *arg, const char *format, ...);
00105
00106 #ifdef __cplusplus
00107 }
00108 #endif
00109
00110 #endif // __PRINTF_H_

```

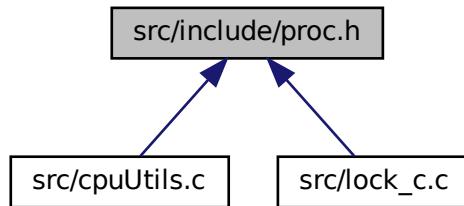
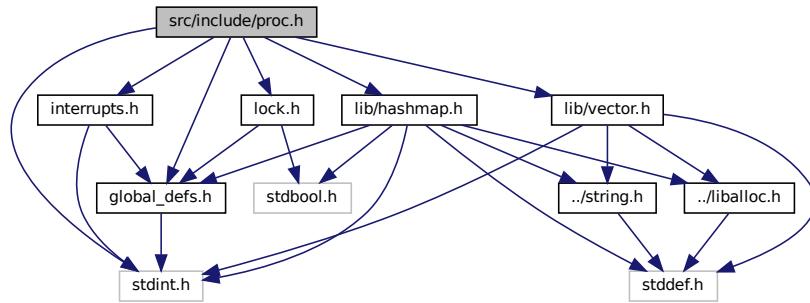
## 4.65 src/include/proc.h File Reference

```

#include <stdint.h>
#include "lock.h"
#include "interrupts.h"
#include "global_defs.h"
#include "lib/hashmap.h"
#include "lib/vector.h"

```

Include dependency graph for proc.h:



## Data Structures

- struct `proc_vmm_map`
- struct `process`

## Functions

- void `proc_yield()`

### 4.65.1 Data Structure Documentation

#### 4.65.1.1 struct `proc_vmm_map`

Definition at line 15 of file `proc.h`.

##### Data Fields

<code>uint64_t</code>	<code>page_count</code>	
	<code>pages</code>	
<small>Generated by Doxygen void *</small>	<code>phys</code>	
<code>void *</code>	<code>virt</code>	

#### 4.65.1.2 struct process

Definition at line 25 of file [proc.h](#).

##### Data Fields

char **	argv	
uint64_t	cr3	
char **	envr	
uint64_t *	ist_stack	
uint64_t *	kernel_stack	
uint64_t	privilege_level	
uint64_t	rflags	
uint64_t	rip	
uint64_t	rsp	
uint64_t	sleep_ticks	
uint64_t *	stack	

#### 4.65.2 Function Documentation

##### 4.65.2.1 proc\_yield()

```
void proc_yield( )
```

#### 4.66 proc.h

```
00001 #pragma once
00002
00003 #include <stdint.h>
00004 // #include <signal.h>
00005 #include "lock.h"
00006 #include "interrupts.h"
00007 #include "global_defs.h"
00008 #include "lib/hashmap.h"
00009 #include "lib/vector.h"
00010
00011 extern void proc_yield();
00012
00013 // const int PROCESS_STACK_SIZE = 0x4000;
00014
00015 struct proc_vmm_map
00016 {
00017
00018     void *virt;
00019     void *phys;
00020     VECTORTYPE(void *)
00021     pages;
00022     uint64_t page_count;
00023 };
00024
00025 struct process
00026 {
00027
00028     // pid_t id;
00029     uint64_t privilege_level;
00030     // char[128] proc_name;
00031     char **argv;
00032     char **envr;
```

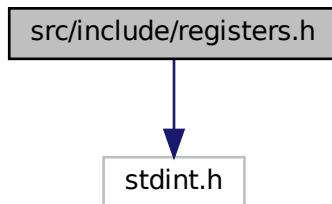
```

00033     uint64_t *kernel_stack;
00034     uint64_t *ist_stack;
00035     uint64_t *stack;
00036     uint64_t rip;
00037     uint64_t rsp;
00038     uint64_t cr3;
00039     uint64_t rflags;
00040     uint64_t sleep_ticks;
00041 // struct sigaction sig_handlers[SIGNAL_MAX];
00042 };

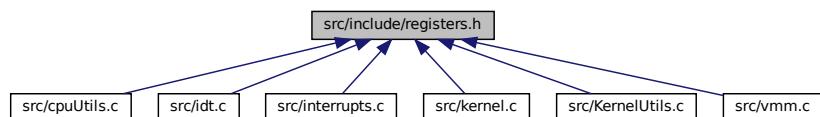
```

## 4.67 src/include/registers.h File Reference

#include <stdint.h>  
 Include dependency graph for registers.h:



This graph shows which files directly or indirectly include this file:



## Functions

- uint64\_t **readCRO** ()
- uint64\_t **readCR3** ()
- uint64\_t **readCR2** ()
- uint64\_t **readRSP** ()
- uint64\_t **readCR4** ()
- void **writeCR4** (uint64\_t rdi)
- void **writeCR0** (uint64\_t rdi)
- void **writeCR3** (uint64\_t rdi)
- void **writeMSR** (uint64\_t msr, uint64\_t value)

## 4.67.1 Function Documentation

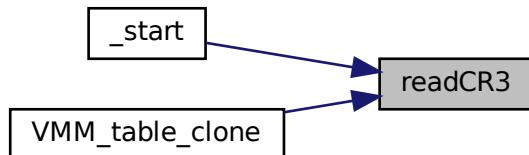
### 4.67.1.1 readCR2()

```
uint64_t readCR2 ( )
```

### 4.67.1.2 readCR3()

```
uint64_t readCR3 ( )
```

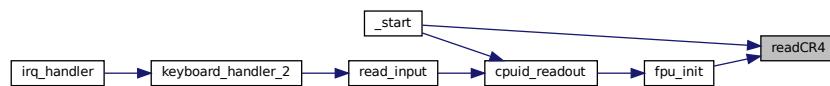
Here is the caller graph for this function:



### 4.67.1.3 readCR4()

```
uint64_t readCR4 ( )
```

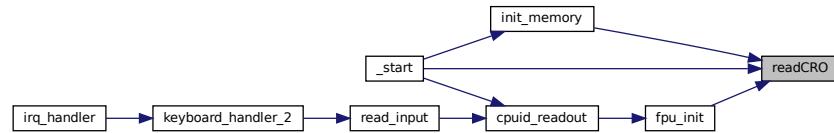
Here is the caller graph for this function:



#### 4.67.1.4 readCRO()

```
uint64_t readCRO ( )
```

Here is the caller graph for this function:



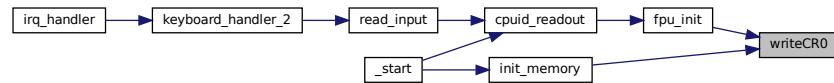
#### 4.67.1.5 readRSP()

```
uint64_t readRSP ( )
```

#### 4.67.1.6 writeCR0()

```
void writeCR0 (
    uint64_t rdi )
```

Here is the caller graph for this function:



#### 4.67.1.7 writeCR3()

```
void writeCR3 (
    uint64_t rdi )
```

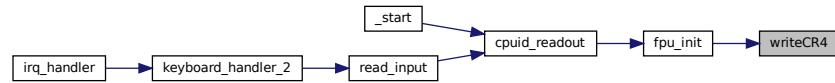
Here is the caller graph for this function:



#### 4.67.1.8 writeCR4()

```
void writeCR4 (
    uint64_t rdi )
```

Here is the caller graph for this function:

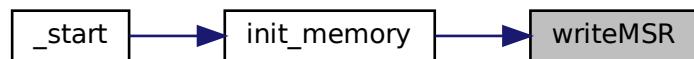


#### 4.67.1.9 writeMSR()

```
void writeMSR (
    uint64_t msr,
    uint64_t value )
```

Definition at line 3 of file [registers.c](#).

Here is the caller graph for this function:

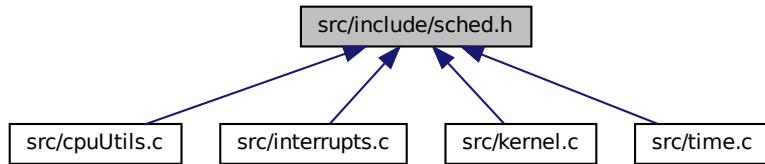


## 4.68 registers.h

```
00001 #pragma once
00002 #include <stdint.h>
00003
00004 extern uint64_t readCRO();
00005 extern uint64_t readCR3();
00006 extern uint64_t readCR2();
00007 extern uint64_t readRSP();
00008 extern uint64_t readCR4();
00009 extern void writeCR4(uint64_t rdi);
00010 extern void writeCRO(uint64_t rdi);
00011 extern void writeCR3(uint64_t rdi);
00012 void writeMSR(uint64_t msr, uint64_t value);
```

## 4.69 src/include/sched.h File Reference

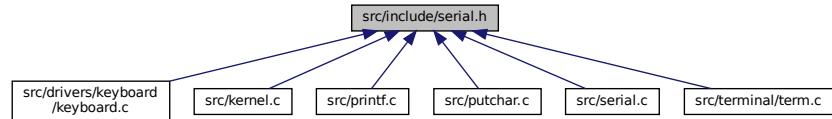
This graph shows which files directly or indirectly include this file:



## 4.70 sched.h

### 4.71 src/include/serial.h File Reference

This graph shows which files directly or indirectly include this file:



## Functions

- void `serial_debug` (int c)
- void `serial_print` (const char \*str)
- void `serial_print_line` (const char \*str)

### 4.71.1 Function Documentation

#### 4.71.1.1 serial\_debug()

```
void serial_debug (
    int c )
```

#### 4.71.1.2 serial\_print()

```
void serial_print (
    const char * str )
```

Definition at line 4 of file [serial.c](#).

Here is the call graph for this function:

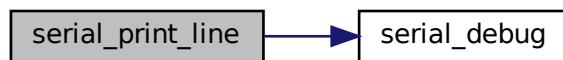


#### 4.71.1.3 serial\_print\_line()

```
void serial_print_line (
    const char * str )
```

Definition at line 14 of file [serial.c](#).

Here is the call graph for this function:

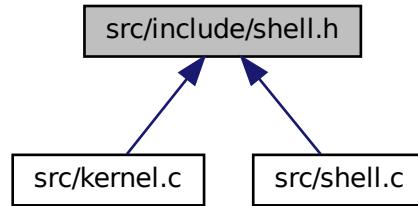


## 4.72 serial.h

```
00001 #ifndef SERIAL_H
00002 #define SERIAL_H
00003
00004 extern void serial_debug(int c);
00005
00006 void serial_print(const char *str);
00007 void serial_print_line(const char *str);
00008
00009 #endif
```

## 4.73 src/include/shell.h File Reference

This graph shows which files directly or indirectly include this file:



### Macros

- #define \_SHELL\_H
- #define VSH\_CMD\_BUFFER\_SIZE 100
- #define VSH\_VERSION "0.0.1"

### Functions

- void vsh\_loop ()
- char vsh\_readline ()

#### 4.73.1 Macro Definition Documentation

##### 4.73.1.1 \_SHELL\_H

```
#define _SHELL_H
```

Definition at line 4 of file shell.h.

##### 4.73.1.2 VSH\_CMD\_BUFFER\_SIZE

```
#define VSH_CMD_BUFFER_SIZE 100
```

Definition at line 6 of file shell.h.

### 4.73.1.3 VSH\_VERSION

```
#define VSH_VERSION "0.0.1"
```

Definition at line 7 of file [shell.h](#).

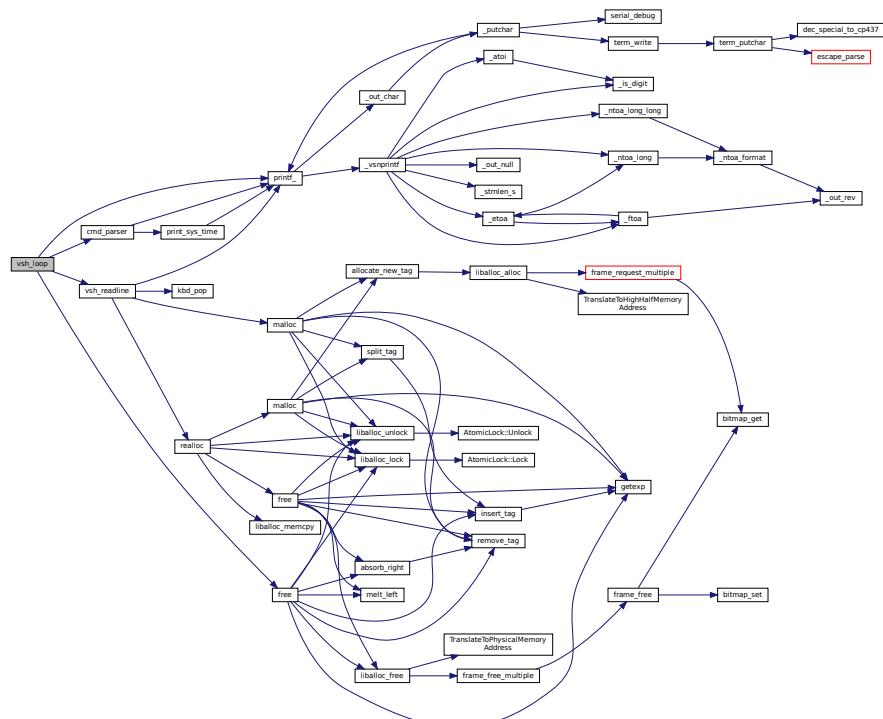
## 4.73.2 Function Documentation

### 4.73.2.1 vsh\_loop()

```
void vsh_loop ( )
```

Definition at line 12 of file [shell.c](#).

Here is the call graph for this function:

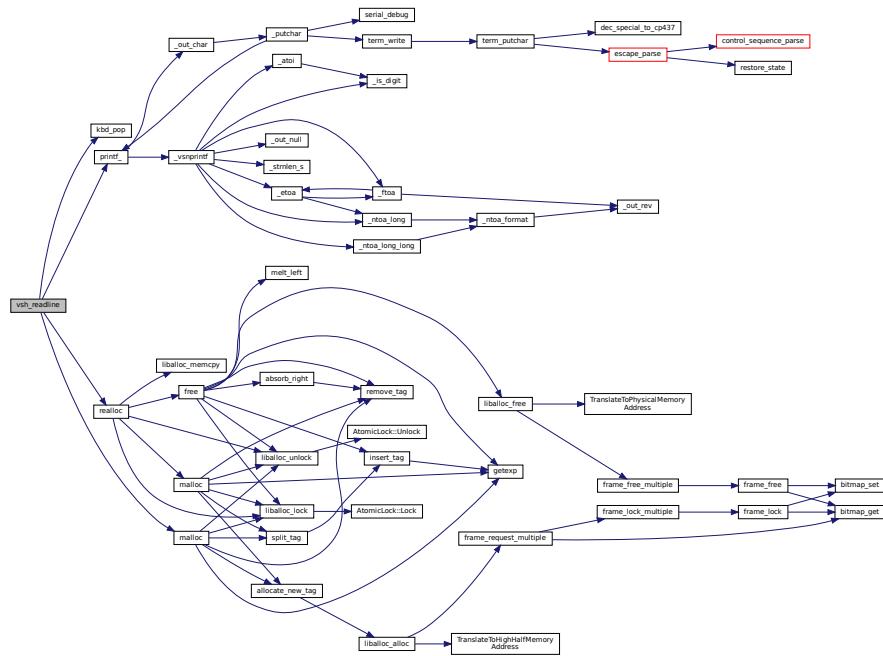


### 4.73.2.2 vsh\_readline()

```
char vsh_readline ( )
```

Definition at line 28 of file [shell.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

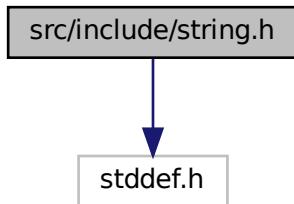


## 4.74 shell.h

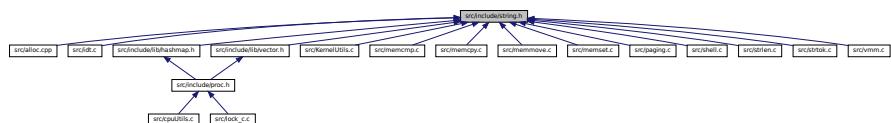
```
00001 #pragma once
00002
00003 ifndef _SHELL_H
00004 define _SHELL_H
00005
00006 define VSH_CMD_BUFFER_SIZE 100
00007 define VSH_VERSION "0.0.1"
00008
00009 void vsh_loop();
00010
00011 char vsh_readline();
00012
00013 endif
```

## 4.75 src/include/string.h File Reference

```
#include <stddef.h>
Include dependency graph for string.h:
```



This graph shows which files directly or indirectly include this file:



## Functions

- int [memcmp](#) (const void \*, const void \*, size\_t)
- void \* [memcpy](#) (void \* \_\_restrict, const void \* \_\_restrict, size\_t)
- void \* [memmove](#) (void \*, const void \*, size\_t)
- void \* [memset](#) (void \*, int, size\_t)
- size\_t [strlen](#) (const char \*)
- int [strtok](#) (char \*srcstr, char sep, char \*\*\*output)

### 4.75.1 Function Documentation

#### 4.75.1.1 memcmp()

```
int memcmp (
    const void * aptr,
    const void * bptr,
    size_t size )
```

Definition at line 3 of file [memcmp.c](#).

### 4.75.1.2 memcpy()

```
void* memcpy (
    void * __restrict,
    const void * __restrict,
    size_t )
```

Here is the caller graph for this function:



### 4.75.1.3 memmove()

```
void* memmove (
    void * dstptr,
    const void * srcptr,
    size_t size )
```

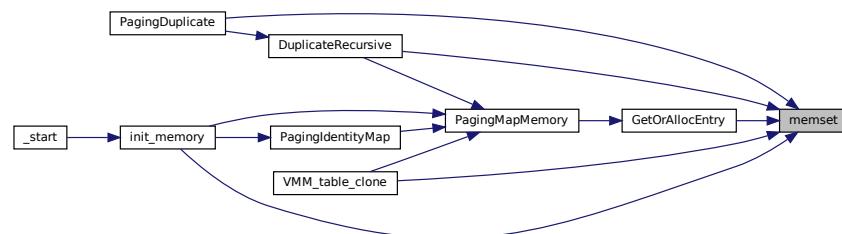
Definition at line 3 of file [memmove.c](#).

### 4.75.1.4 memset()

```
void* memset (
    void * bufptr,
    int value,
    size_t size )
```

Definition at line 3 of file [memset.c](#).

Here is the caller graph for this function:



#### 4.75.1.5 `strlen()`

```
size_t strlen (
    const char * str )
```

Definition at line 3 of file [strlen.c](#).

Here is the caller graph for this function:

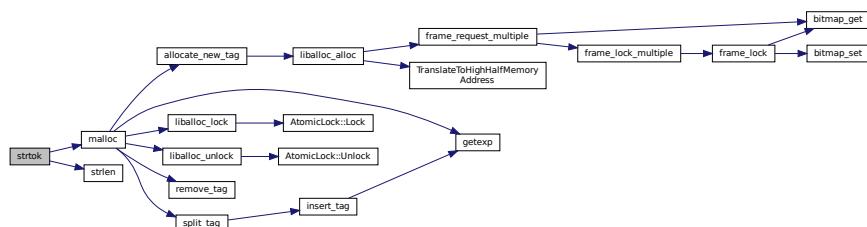


#### 4.75.1.6 `strtok()`

```
int strtok (
    char * srcstr,
    char sep,
    char *** output )
```

Definition at line 5 of file [strtok.c](#).

Here is the call graph for this function:



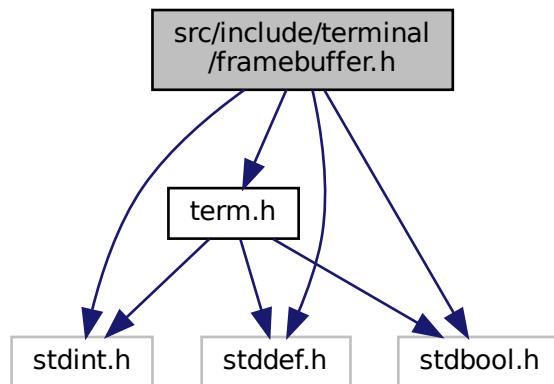
## 4.76 string.h

```
00001 #ifndef _STRING_H
00002 #define _STRING_H 1
00003
00004 #include <stddef.h>
00005
00006 #ifdef __cplusplus
00007 extern "C"
00008 {
00009 #endif
0010
0011     int memcmp(const void *, const void *, size_t);
0012     void *memcpy(void *__restrict, const void *__restrict, size_t);
```

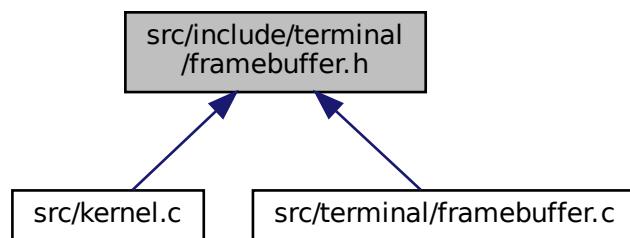
```
00013     void *memmove(void *, const void *, size_t);
00014     void *memset(void *, int, size_t);
00015     size_t strlen(const char *);
00016     int strtok(char *srcstr, char sep, char ***output);
00017
00018 #ifdef __cplusplus
00019 }
00020 #endif
00021
00022 #endif
```

## 4.77 src/include/terminal/framebuffer.h File Reference

```
#include <stdint.h>
#include <stddef.h>
#include <stdbool.h>
#include "term.h"
Include dependency graph for framebuffer.h:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `fbterm_char`
- struct `fbterm_queue_item`
- struct `fbterm_context`

## Macros

- `#define FBTERM_FONT_GLYPHS 256`

## Functions

- struct `term_context * fbterm_init` (void `*(_malloc)(size_t)`, `uint32_t *framebuffer`, `size_t width`, `size_t height`, `size_t pitch`, `uint32_t *canvas`, `uint32_t *ansi_colours`, `uint32_t *ansi_bright_colours`, `uint32_t *default_bg`, `uint32_t *default_fg`, `void *font`, `size_t font_width`, `size_t font_height`, `size_t font_spacing`, `size_t font_scale_x`, `size_t font_scale_y`, `size_t margin`)

### 4.77.1 Data Structure Documentation

#### 4.77.1.1 struct `fbterm_char`

Definition at line 12 of file `framebuffer.h`.

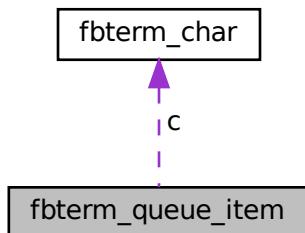
##### Data Fields

<code>uint32_t</code>	<code>bg</code>	
<code>uint32_t</code>	<code>c</code>	
<code>uint32_t</code>	<code>fg</code>	

#### 4.77.1.2 struct `fbterm_queue_item`

Definition at line 19 of file `framebuffer.h`.

Collaboration diagram for `fbterm_queue_item`:



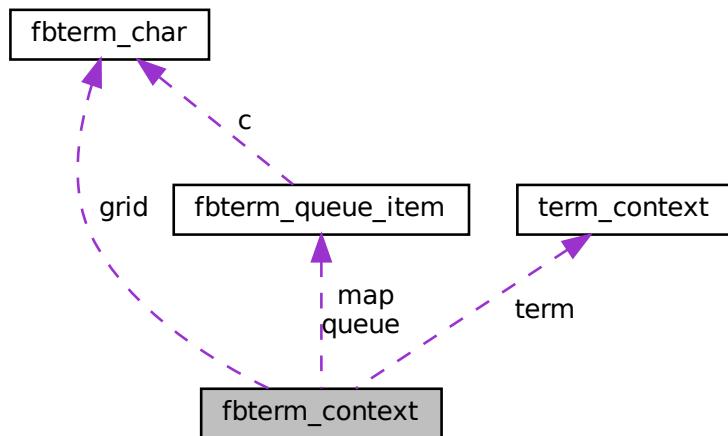
## Data Fields

struct <b>fbterm_char</b>	c	
	size_t	x
	size_t	y

4.77.1.3 struct **fbterm\_context**

Definition at line 25 of file [framebuffer.h](#).

Collaboration diagram for **fbterm\_context**:



## Data Fields

uint32_t	ansi_bright_colours[8]	
uint32_t	ansi_colours[8]	
size_t	bpp	
uint32_t *	canvas	
size_t	canvas_size	
bool	cursor_status	
size_t	cursor_x	
size_t	cursor_y	
uint32_t	default_bg	
uint32_t	default_fg	
uint8_t *	font_bits	
size_t	font_bits_size	
bool *	font_bool	
size_t	font_bool_size	
size_t	font_height	
size_t	font_scale_x	

## Data Fields

size_t	font_scale_y	
size_t	font_width	
volatile uint32_t *	framebuffer	
size_t	glyph_height	
size_t	glyph_width	
struct fbterm_char *	grid	
size_t	grid_size	
size_t	height	
struct fbterm_queue_item **	map	
size_t	map_size	
size_t	offset_x	
size_t	offset_y	
size_t	old_cursor_x	
size_t	old_cursor_y	
size_t	pitch	
struct fbterm_queue_item *	queue	
size_t	queue_i	
size_t	queue_size	
size_t	saved_state_cursor_x	
size_t	saved_state_cursor_y	
uint32_t	saved_state_text_bg	
uint32_t	saved_state_text_fg	
struct term_context	term	
uint32_t	text_bg	
uint32_t	text_fg	
size_t	width	

**4.77.2 Macro Definition Documentation****4.77.2.1 FBTERM\_FONT\_GLYPHS**

```
#define FBTERM_FONT_GLYPHS 256
```

Definition at line 10 of file [framebuffer.h](#).

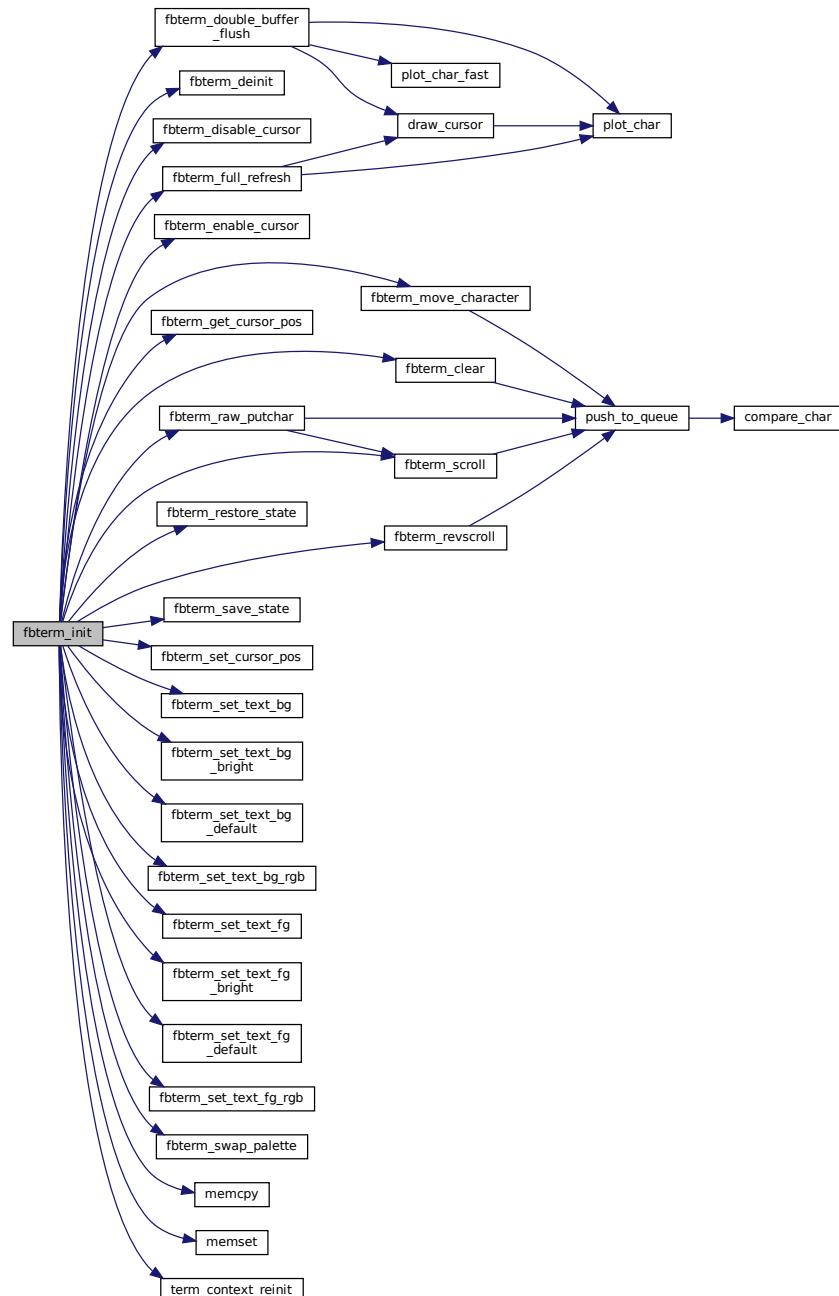
**4.77.3 Function Documentation**

#### 4.77.3.1 fbterm\_init()

```
struct term_context* fbterm_init (
    void *(*)(size_t) _malloc,
    uint32_t * framebuffer,
    size_t width,
    size_t height,
    size_t pitch,
    uint32_t * canvas,
    uint32_t * ansi_colours,
    uint32_t * ansi_bright_colours,
    uint32_t * default_bg,
    uint32_t * default_fg,
    void * font,
    size_t font_width,
    size_t font_height,
    size_t font_spacing,
    size_t font_scale_x,
    size_t font_scale_y,
    size_t margin )
```

Definition at line 668 of file [framebuffer.c](#).

Here is the call graph for this function:



## 4.78 framebuffer.h

```

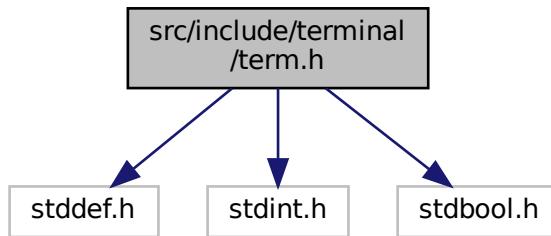
00001 #ifndef _TERM_FRAMEBUFFER_H
00002 #define _TERM_FRAMEBUFFER_H
00003
00004 #include <stdint.h>
00005 #include <stddef.h>
00006 #include <stdbool.h>
00007
00008 #include "term.h"
00009
00100 #define FBTERM_FONT_GLYPHS 256

```

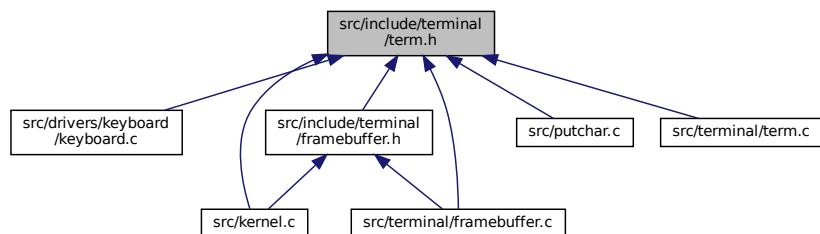
```
00011
00012 struct fbterm_char
00013 {
00014     uint32_t c;
00015     uint32_t fg;
00016     uint32_t bg;
00017 };
00018
00019 struct fbterm_queue_item
00020 {
00021     size_t x, y;
00022     struct fbterm_char c;
00023 };
00024
00025 struct fbterm_context
00026 {
00027     struct term_context term;
00028
00029     size_t font_width;
00030     size_t font_height;
00031     size_t glyph_width;
00032     size_t glyph_height;
00033
00034     size_t font_scale_x;
00035     size_t font_scale_y;
00036
00037     size_t offset_x, offset_y;
00038
00039     volatile uint32_t *framebuffer;
00040     size_t pitch;
00041     size_t width;
00042     size_t height;
00043     size_t bpp;
00044
00045     size_t font_bits_size;
00046     uint8_t *font_bits;
00047     size_t font_bool_size;
00048     bool *font_bool;
00049
00050     uint32_t ansi_colours[8];
00051     uint32_t ansi_bright_colours[8];
00052     uint32_t default_fg, default_bg;
00053
00054     size_t canvas_size;
00055     uint32_t *canvas;
00056
00057     size_t grid_size;
00058     size_t queue_size;
00059     size_t map_size;
00060
00061     struct fbterm_char *grid;
00062
00063     struct fbterm_queue_item *queue;
00064     size_t queue_i;
00065
00066     struct fbterm_queue_item **map;
00067
00068     uint32_t text_fg;
00069     uint32_t text_bg;
00070     bool cursor_status;
00071     size_t cursor_x;
00072     size_t cursor_y;
00073
00074     uint32_t saved_state_text_fg;
00075     uint32_t saved_state_text_bg;
00076     size_t saved_state_cursor_x;
00077     size_t saved_state_cursor_y;
00078
00079     size_t old_cursor_x;
00080     size_t old_cursor_y;
00081 };
00082
00083 struct term_context *fbterm_init(
00084     void *(*_malloc)(size_t),
00085     uint32_t *framebuffer, size_t width, size_t height, size_t pitch,
00086     uint32_t *canvas,
00087     uint32_t *ansi_colours, uint32_t *ansi_bright_colours,
00088     uint32_t *default_bg, uint32_t *default_fg,
00089     void *font, size_t font_width, size_t font_height, size_t font_spacing,
00090     size_t font_scale_x, size_t font_scale_y,
00091     size_t margin);
00092
00093 #endif
```

## 4.79 src/include/terminal/term.h File Reference

```
#include <stddef.h>
#include <stdint.h>
#include <stdbool.h>
Include dependency graph for term.h:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [term\\_context](#)

## Macros

- #define TERM\_MAX\_ESC\_VALUES 16
- #define TERM\_CB\_DEC 10
- #define TERM\_CB\_BELL 20
- #define TERM\_CB\_PRIVATE\_ID 30
- #define TERM\_CB\_STATUS\_REPORT 40
- #define TERM\_CB\_POS\_REPORT 50
- #define TERM\_CB\_KBD\_LEDS 60
- #define TERM\_CB\_MODE 70
- #define TERM\_CB\_LINUX 80

## Functions

- void `term_context_reinit` (struct `term_context` \*ctx)
- void `term_write` (struct `term_context` \*ctx, const char \*buf, size\_t count)

### 4.79.1 Macro Definition Documentation

#### 4.79.1.1 TERM\_CB\_BELL

```
#define TERM_CB_BELL 20
```

Definition at line 11 of file [term.h](#).

#### 4.79.1.2 TERM\_CB\_DEC

```
#define TERM_CB_DEC 10
```

Definition at line 10 of file [term.h](#).

#### 4.79.1.3 TERM\_CB\_KBD\_LEDS

```
#define TERM_CB_KBD_LEDS 60
```

Definition at line 15 of file [term.h](#).

#### 4.79.1.4 TERM\_CB\_LINUX

```
#define TERM_CB_LINUX 80
```

Definition at line 17 of file [term.h](#).

#### 4.79.1.5 TERM\_CB\_MODE

```
#define TERM_CB_MODE 70
```

Definition at line 16 of file [term.h](#).

#### 4.79.1.6 TERM\_CB\_POS\_REPORT

```
#define TERM_CB_POS_REPORT 50
```

Definition at line [14](#) of file [term.h](#).

#### 4.79.1.7 TERM\_CB\_PRIVATE\_ID

```
#define TERM_CB_PRIVATE_ID 30
```

Definition at line [12](#) of file [term.h](#).

#### 4.79.1.8 TERM\_CB\_STATUS\_REPORT

```
#define TERM_CB_STATUS_REPORT 40
```

Definition at line [13](#) of file [term.h](#).

#### 4.79.1.9 TERM\_MAX\_ESC\_VALUES

```
#define TERM_MAX_ESC_VALUES 16
```

Definition at line [8](#) of file [term.h](#).

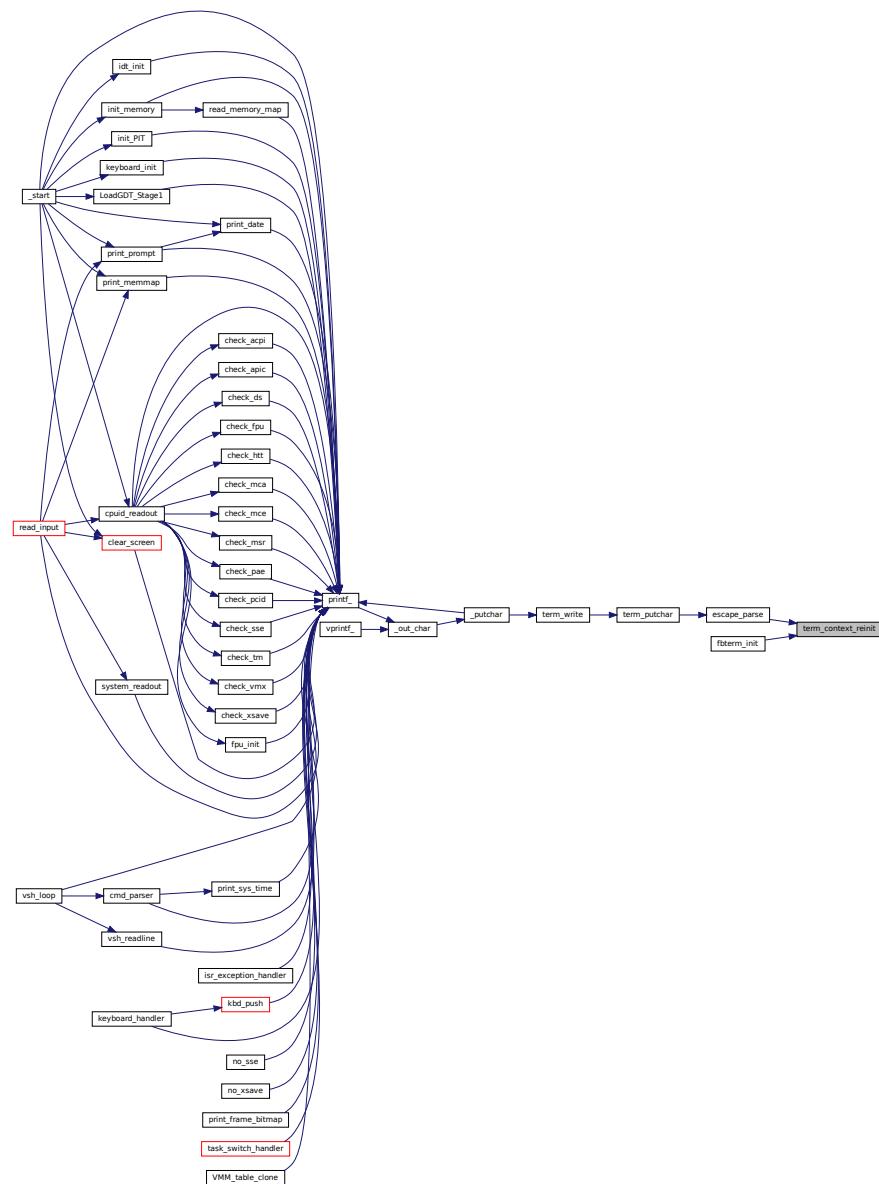
### 4.79.2 Function Documentation

### 4.79.2.1 term\_context\_reinit()

```
void term_context_reinit (
    struct term_context * ctx )
```

Definition at line 46 of file [term.c](#).

Here is the caller graph for this function:



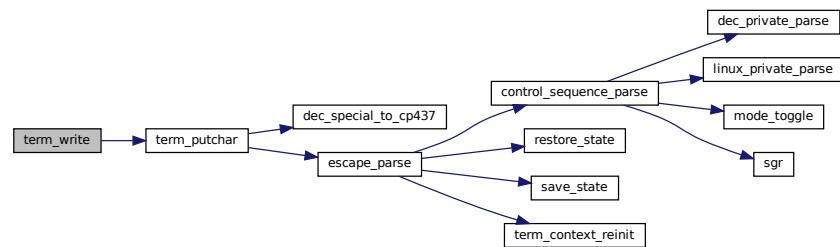
### 4.79.2.2 term\_write()

```
void term_write (
    struct term_context * ctx,
```

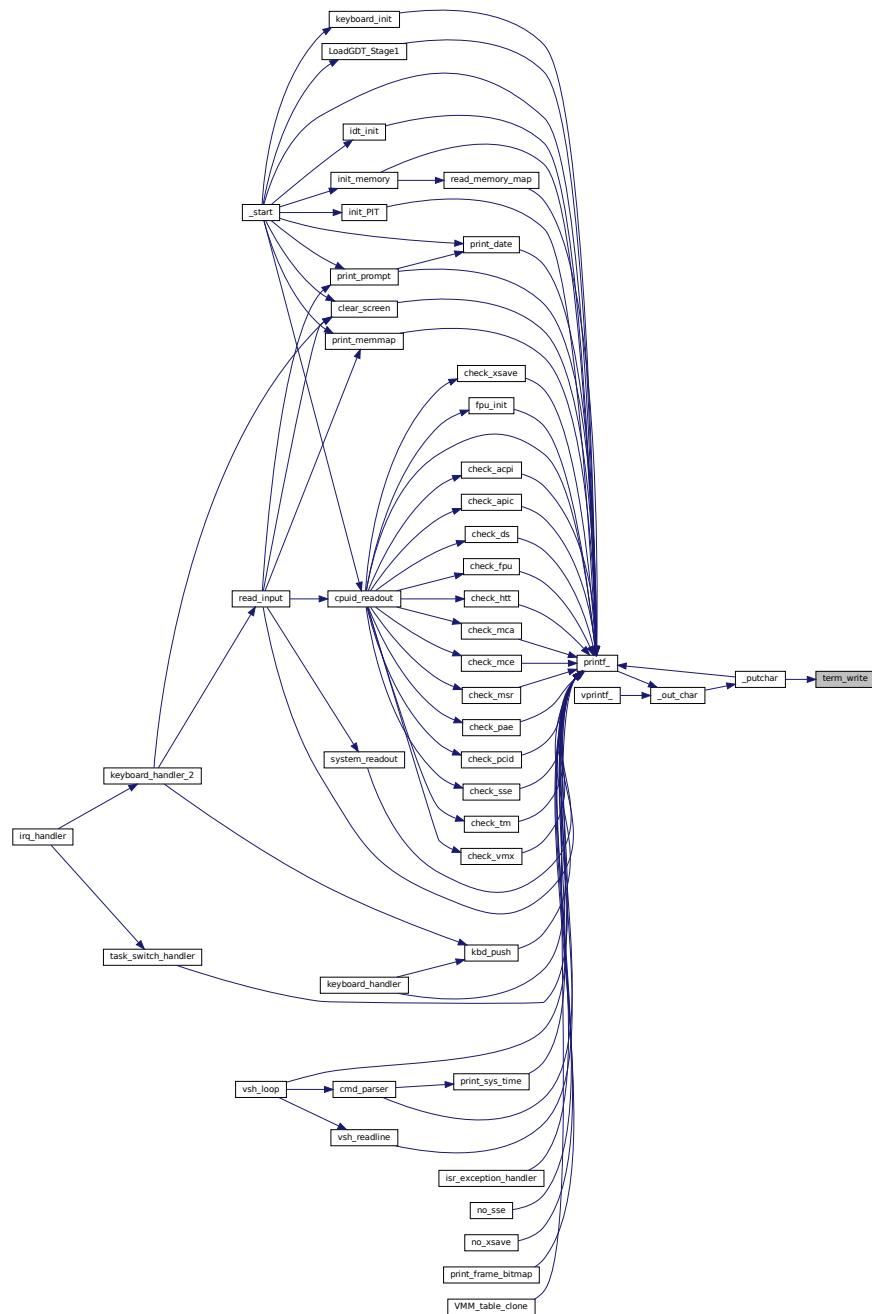
```
const char * buf,  
size_t count )
```

Definition at line 75 of file [term.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.80 term.h

```

00001 #ifndef _TERM_H
00002 #define _TERM_H
00003
00004 #include <stddef.h>
00005 #include <stdint.h>
00006 #include <stdbool.h>
00007
00008 #define TERM_MAX_ESC_VALUES 16
00009
00010 #define TERM_CB_DEC 10
00011 #define TERM_CB_BELL 20
00012 #define TERM_CB_PRIVATE_ID 30

```

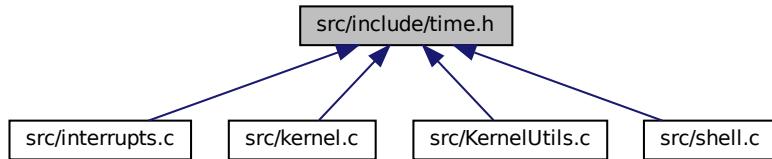
```

00013 #define TERM_CB_STATUS_REPORT 40
00014 #define TERM_CB_POS_REPORT 50
00015 #define TERM_CB_KBD_LEDS 60
00016 #define TERM_CB_MODE 70
00017 #define TERM_CB_LINUX 80
00018
00019 struct term_context
00020 {
00021     /* internal use */
00022
00023     size_t tab_size;
00024     bool autoflush;
00025     bool scroll_enabled;
00026     bool control_sequence;
00027     bool csi;
00028     bool escape;
00029     bool rrr;
00030     bool discard_next;
00031     bool bold;
00032     bool reverse_video;
00033     bool dec_private;
00034     bool insert_mode;
00035     uint8_t g_select;
00036     uint8_t charsets[2];
00037     size_t current_charset;
00038     size_t escape_offset;
00039     size_t esc_values_i;
00040     size_t saved_cursor_x;
00041     size_t saved_cursor_y;
00042     size_t current_primary;
00043     size_t scroll_top_margin;
00044     size_t scroll_bottom_margin;
00045     uint32_t esc_values[TERM_MAX_ESC_VALUES];
00046     bool saved_state_bold;
00047     bool saved_state_reverse_video;
00048     size_t saved_state_current_charset;
00049     size_t saved_state_current_primary;
00050
00051     /* to be set by backend */
00052
00053     size_t rows, cols;
00054     bool in_bootloader;
00055
00056     void (*raw_putchar)(struct term_context *, uint8_t c);
00057     void (*clear)(struct term_context *, bool move);
00058     void (*enable_cursor)(struct term_context *);
00059     bool (*disable_cursor)(struct term_context *);
00060     void (*set_cursor_pos)(struct term_context *, size_t x, size_t y);
00061     void (*get_cursor_pos)(struct term_context *, size_t *x, size_t *y);
00062     void (*set_text_fg)(struct term_context *, size_t fg);
00063     void (*set_text_bg)(struct term_context *, size_t bg);
00064     void (*set_text_fg_bright)(struct term_context *, size_t fg);
00065     void (*set_text_bg_bright)(struct term_context *, size_t bg);
00066     void (*set_text_fg_rgb)(struct term_context *, uint32_t fg);
00067     void (*set_text_bg_rgb)(struct term_context *, uint32_t bg);
00068     void (*set_text_fg_default)(struct term_context *);
00069     void (*set_text_bg_default)(struct term_context *);
00070     void (*move_character)(struct term_context *, size_t new_x, size_t new_y, size_t old_x, size_t
old_y);
00071     void (*scroll)(struct term_context *);
00072     void (*revscroll)(struct term_context *);
00073     void (*swap_palette)(struct term_context *);
00074     void (*save_state)(struct term_context *);
00075     void (*restore_state)(struct term_context *);
00076     void (*double_buffer_flush)(struct term_context *);
00077     void (*full_refresh)(struct term_context *);
00078     void (*deinit)(struct term_context *, void (*) (void *, size_t));
00079
00080     /* to be set by client */
00081
00082     void (*callback)(struct term_context *, uint64_t, uint64_t, uint64_t, uint64_t);
00083 };
00084
00085 void term_context_reinit(struct term_context *ctx);
00086 void term_write(struct term_context *ctx, const char *buf, size_t count);
00087
00088 #endif

```

## 4.81 src/include/time.h File Reference

This graph shows which files directly or indirectly include this file:



### Macros

- `#define _TIME_H`

### Functions

- `void sys_clock_handler ()`
- `void init_PIT ()`
- `void read_rtc ()`
- `void print_date ()`
- `void print_sys_time ()`
- `void sleep (uint64_t millis)`

### Variables

- `uint64_t system_timer_ms`
- `uint64_t system_timer_fractions`
- `uint8_t second`
- `uint8_t minute`
- `uint8_t hour`
- `uint8_t day`
- `uint8_t month`
- `uint8_t year`
- `uint64_t pit_armed`

### 4.81.1 Macro Definition Documentation

#### 4.81.1.1 \_TIME\_H

```
#define _TIME_H
```

Definition at line 4 of file [time.h](#).

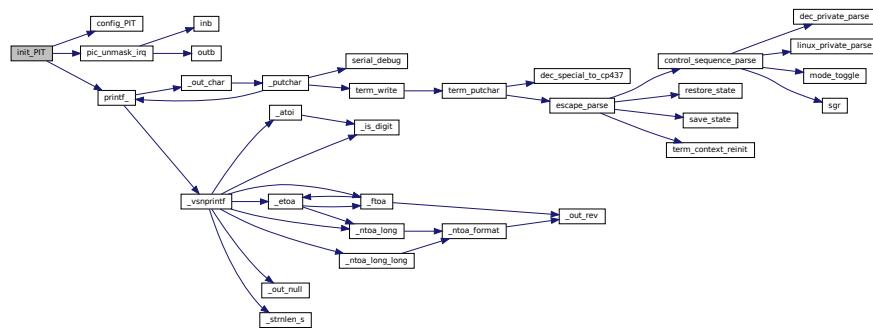
## 4.81.2 Function Documentation

### 4.81.2.1 init\_PIT()

```
void init_PIT ( )
```

Definition at line 104 of file [time.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

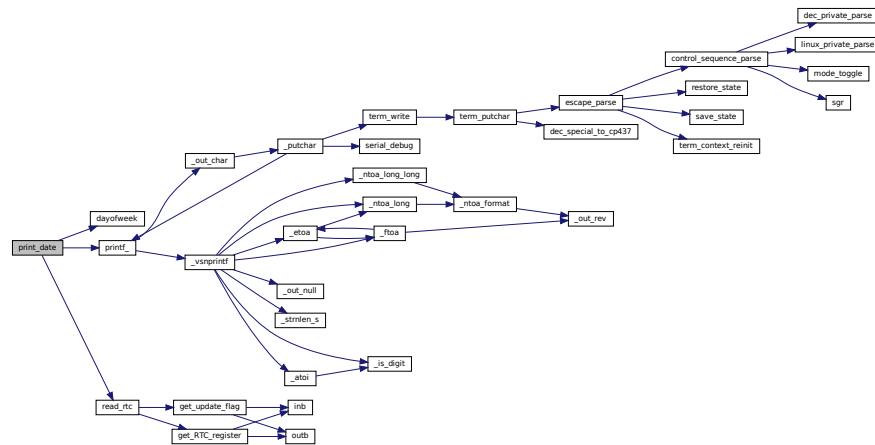


### 4.81.2.2 print\_date()

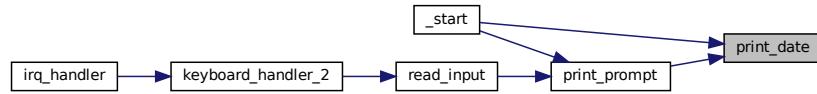
```
void print_date ( )
```

Definition at line 223 of file [time.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

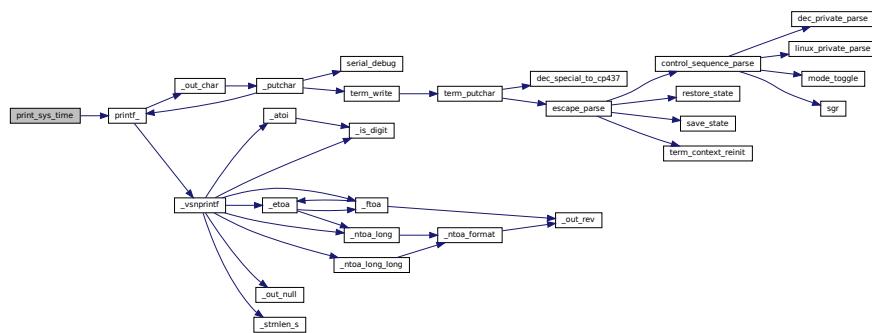


### 4.81.2.3 print\_sys\_time()

```
void print_sys_time ( )
```

Definition at line 259 of file [time.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

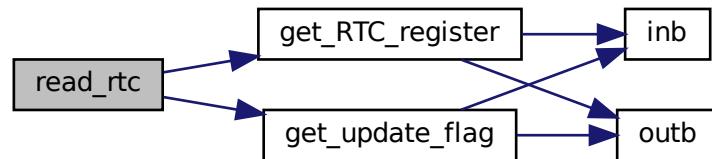


#### 4.81.2.4 read\_rtc()

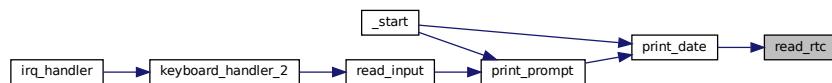
```
void read_rtc ( )
```

Definition at line 130 of file [time.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.81.2.5 sleep()

```
void sleep (
    uint64_t millis )
```

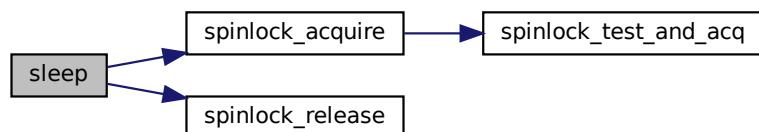
Sleep for a number of milliseconds. This is useful for unit testing to ensure that the program doesn't die indefinitely

**Parameters**

<code>millis</code>	- The number of milliseconds to
---------------------	---------------------------------

Definition at line 64 of file [time.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.81.2.6 sys\_clock\_handler()

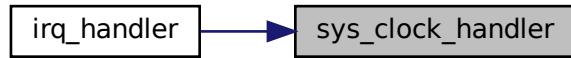
```
void sys_clock_handler ( )
```

Definition at line 81 of file [time.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 4.81.3 Variable Documentation

#### 4.81.3.1 day

```
uint8_t day
```

Definition at line 10 of file [time.h](#).

#### 4.81.3.2 hour

```
uint8_t hour
```

Definition at line 10 of file [time.h](#).

#### 4.81.3.3 minute

```
uint8_t minute
```

Definition at line 10 of file [time.h](#).

#### 4.81.3.4 month

```
uint8_t month
```

Definition at line 10 of file [time.h](#).

#### 4.81.3.5 pit\_armed

```
uint64_t pit_armed [extern]
```

Definition at line 28 of file [time.c](#).

#### 4.81.3.6 second

```
uint8_t second [extern]
```

Definition at line 20 of file [time.c](#).

#### 4.81.3.7 system\_timer\_fractions

```
uint64_t system_timer_fractions [extern]
```

Definition at line 24 of file [time.c](#).

#### 4.81.3.8 system\_timer\_ms

```
uint64_t system_timer_ms [extern]
```

Definition at line 22 of file [time.c](#).

#### 4.81.3.9 year

```
uint8_t year
```

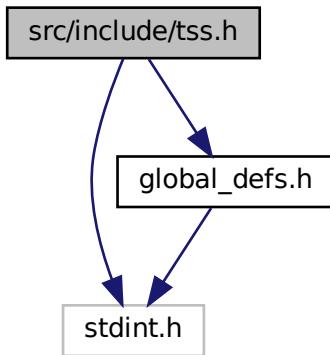
Definition at line 10 of file [time.h](#).

## 4.82 time.h

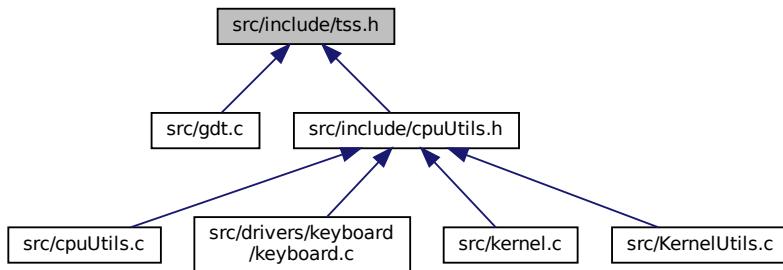
```
00001 #pragma once
00002
00003 #ifndef _TIME_H
00004 #define _TIME_H
00005
00006 extern uint64_t system_timer_ms;
00007
00008 extern uint64_t system_timer_fractions;
00009
00010 extern uint8_t second, minute, hour, day, month, year;
00011
00012 extern uint64_t pit_armed;
00013
00014 void sys_clock_handler();
00015
00016 void init_PIT();
00017
00018 void read_rtc();
00019
00020 void print_date();
00021
00022 void print_sys_time();
00023
00024 void sleep(uint64_t millis);
00025
00026 #endif
```

## 4.83 src/include/tss.h File Reference

```
#include <stdint.h>
#include "global_defs.h"
Include dependency graph for tss.h:
```



This graph shows which files directly or indirectly include this file:



### Data Structures

- struct [TSS](#)

### Macros

- #define [\\_TSS\\_H](#)

#### 4.83.1 Data Structure Documentation

##### 4.83.1.1 struct TSS

Definition at line [8](#) of file [tss.h](#).

### Data Fields

uint16_t	iopbOffset	
uint64_t	ist1	
uint64_t	ist2	
uint64_t	ist3	
uint64_t	ist4	
uint64_t	ist5	
uint64_t	ist6	
uint64_t	ist7	
uint32_t	reserved0	
uint64_t	reserved1	
uint64_t	reserved2	
uint16_t	reserved3	
uint64_t	rsp0	
uint64_t	rsp1	
uint64_t	rsp2	

## 4.83.2 Macro Definition Documentation

### 4.83.2.1 \_TSS\_H

```
#define _TSS_H
```

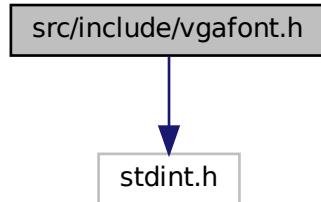
Definition at line 6 of file [tss.h](#).

## 4.84 tss.h

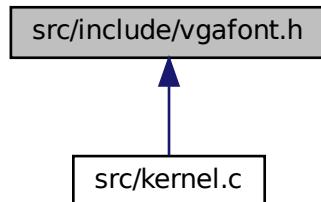
```
00001 #pragma once
00002 #include <stdint.h>
00003 #include "global_defs.h"
00004
00005 #ifndef _TSS_H
00006 #define _TSS_H
00007
00008 struct PACKED TSS
00009 {
00010     uint32_t reserved0;
00011     uint64_t rsp0;
00012     uint64_t rsp1;
00013     uint64_t rsp2;
00014     uint64_t reserved1;
00015     uint64_t ist1;
00016     uint64_t ist2;
00017     uint64_t ist3;
00018     uint64_t ist4;
00019     uint64_t ist5;
00020     uint64_t ist6;
00021     uint64_t ist7;
00022     uint64_t reserved2;
00023     uint16_t reserved3;
00024     uint16_t iopbOffset;
00025 };
00026
00027 #endif
```

## 4.85 src/include/vgafont.h File Reference

```
#include <stdint.h>
Include dependency graph for vgafont.h:
```



This graph shows which files directly or indirectly include this file:



### Variables

- static const uint8\_t [vgafont](#) []

#### 4.85.1 Variable Documentation

##### 4.85.1.1 [vgafont](#)

```
const uint8_t vgafont[] [static]
```

Definition at line [6](#) of file [vgafont.h](#).

## 4.86 vgafont.h

```

00001 #ifndef VGAFONT_H
00002 #define VGAFONT_H
00003
00004 #include <stdint.h>
00005
00006 static const uint8_t vgafont[] = {
00007     0x00, 0x00,
00008     0x00, 0x00, 0x7e, 0x81, 0xa5, 0x81, 0x81, 0xbd, 0x99, 0x81, 0x81, 0x7e, 0x00, 0x00, 0x00, 0x00,
00009     0x00, 0x00, 0x7e, 0xff, 0xdb, 0xff, 0xc3, 0x7e, 0xff, 0xff, 0x7e, 0x00, 0x00, 0x00, 0x00,
00010     0x00, 0x00, 0x00, 0x6c, 0xfe, 0xfe, 0xfe, 0x7c, 0x38, 0x7c, 0x38, 0x10, 0x00, 0x00, 0x00, 0x00,
00011     0x00, 0x00, 0x00, 0x10, 0x38, 0x7c, 0xfe, 0x7c, 0x38, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00,
00012     0x00, 0x00, 0x00, 0x18, 0x3c, 0x3c, 0xe7, 0xe7, 0x18, 0x18, 0x3c, 0x00, 0x00, 0x00, 0x00,
00013     0x00, 0x00, 0x00, 0x18, 0x3c, 0x7e, 0xff, 0xff, 0x7e, 0x18, 0x18, 0x3c, 0x00, 0x00, 0x00, 0x00,
00014     0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x3c, 0x3c, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00015     0xf, 0xff, 0xff, 0xff, 0xff, 0x7e, 0x3c, 0x3c, 0x7e, 0x1f, 0xff, 0x7f, 0x7f, 0x7f, 0x7f,
00016     0x00, 0x00, 0x00, 0x00, 0x3c, 0x66, 0x42, 0x42, 0x66, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00,
00017     0xff, 0xff, 0xff, 0xff, 0xc3, 0x99, 0xbd, 0xbd, 0x99, 0xc3, 0xff, 0xff, 0x7f, 0x7f, 0x7f, 0x7f,
00018     0x00, 0x00, 0x1e, 0x0a, 0x1a, 0x32, 0x78, 0xcc, 0xcc, 0xcc, 0x78, 0x00, 0x00, 0x00, 0x00,
00019     0x00, 0x00, 0x3c, 0x66, 0x66, 0x66, 0x3c, 0x18, 0x7e, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00,
00020     0x00, 0x00, 0x3f, 0x33, 0x3f, 0x30, 0x30, 0x30, 0x30, 0x70, 0xf0, 0xe0, 0x00, 0x00, 0x00, 0x00,
00021     0x00, 0x00, 0x7f, 0x63, 0x7f, 0x63, 0x63, 0x63, 0x67, 0xe6, 0xc0, 0x00, 0x00, 0x00, 0x00,
00022     0x00, 0x00, 0x00, 0x18, 0xdb, 0x3c, 0x7e, 0x3c, 0x66, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00,
00023     0x00, 0x80, 0xc0, 0xe0, 0xf0, 0xf8, 0xf0, 0xe0, 0xc0, 0x80, 0x00, 0x00, 0x00, 0x00,
00024     0x00, 0x02, 0x06, 0x0e, 0x1e, 0x3e, 0xfe, 0x3e, 0x1e, 0x0e, 0x06, 0x02, 0x00, 0x00, 0x00, 0x00,
00025     0x00, 0x00, 0x18, 0x3c, 0x7e, 0x18, 0x18, 0x7e, 0x3c, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00,
00026     0x00, 0x00, 0x66, 0x00, 0x00, 0x00, 0x00,
00027     0x00, 0x00, 0x7f, 0xdb, 0xdb, 0x7b, 0x1b, 0x1b, 0x1b, 0x1b, 0x00, 0x00, 0x00, 0x00, 0x00,
00028     0x00, 0x7c, 0xc6, 0x60, 0x38, 0x6c, 0xc6, 0xc6, 0x6c, 0x38, 0x0c, 0x6c, 0x7c, 0x00, 0x00, 0x00,
00029     0x00, 0x00,
00030     0x00, 0x00, 0x18, 0x3c, 0x7e, 0x18, 0x18, 0x7e, 0x3c, 0x18, 0x7e, 0x00, 0x00, 0x00, 0x00, 0x00,
00031     0x00, 0x00, 0x18, 0x3c, 0x7e, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00,
00032     0x00, 0x00, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x7e, 0x3c, 0x18, 0x00, 0x00, 0x00, 0x00,
00033     0x00, 0x00,
00034     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x60, 0xfe, 0x60, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00,
00035     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0, 0xc0, 0x0fe, 0x0fe, 0x00, 0x00, 0x00, 0x00,
00036     0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x66, 0xff, 0x66, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00037     0x00, 0x00, 0x00, 0x00, 0x10, 0x38, 0x38, 0x7c, 0x7c, 0xfe, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00,
00038     0x00, 0x00, 0x00, 0x00, 0xfe, 0xfe, 0x7c, 0x7c, 0x38, 0x38, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00,
00039     0x00, 0x00,
00040     0x00, 0x00, 0x18, 0x3c, 0x3c, 0x3c, 0x18, 0x18, 0x18, 0x00, 0x00, 0x18, 0x18, 0x18, 0x00, 0x00, 0x00,
00041     0x00, 0x66, 0x66, 0x24, 0x00, 0x00,
00042     0x00, 0x00, 0x00, 0x6c, 0x6c, 0xfe, 0x6c, 0x6c, 0x6c, 0x6c, 0xfe, 0x6c, 0x6c, 0x00, 0x00, 0x00, 0x00,
00043     0x18, 0x18, 0x7c, 0xc6, 0xc2, 0xco, 0x7c, 0x06, 0x06, 0x86, 0xc6, 0x7c, 0x18, 0x18, 0x00, 0x00,
00044     0x00, 0x00, 0x00, 0x00, 0x00, 0x2c, 0xc6, 0x0c, 0x18, 0x30, 0x60, 0x6c, 0x86, 0x00, 0x00, 0x00, 0x00,
00045     0x00, 0x00, 0x38, 0x6c, 0x6c, 0x38, 0x76, 0xdc, 0xcc, 0xcc, 0xcc, 0x76, 0x00, 0x00, 0x00, 0x00,
00046     0x00, 0x30, 0x30, 0x30, 0x60, 0x00, 0x00,
00047     0x00, 0x00, 0x0c, 0x18, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x18, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00,
00048     0x00, 0x00, 0x30, 0x18, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x18, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00049     0x00, 0x00, 0x00, 0x00, 0x00, 0x66, 0x3c, 0xff, 0x3c, 0x66, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00050     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x7e, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00051     0x00, 0x18, 0x18, 0x30, 0x00, 0x00, 0x00, 0x00,
00052     0x00, 0x00,
00053     0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00,
00054     0x00, 0x00, 0x00, 0x00, 0x02, 0x06, 0x0c, 0x18, 0x30, 0x60, 0xc0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00,
00055     0x00, 0x00, 0x00, 0x3c, 0x66, 0xc3, 0xc3, 0x66, 0xdb, 0xdb, 0xc3, 0x66, 0x3c, 0x00, 0x00, 0x00, 0x00,
00056     0x00, 0x00, 0x18, 0x38, 0x78, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x7e, 0x00, 0x00, 0x00, 0x00, 0x00,
00057     0x00, 0x00, 0x7c, 0xc6, 0x06, 0x0c, 0x18, 0x30, 0x60, 0xc0, 0x6c, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00,
00058     0x00, 0x00, 0x7c, 0xc6, 0x06, 0x0c, 0x3c, 0x06, 0x0c, 0x6c, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
00059     0x00, 0x00, 0x0c, 0x1c, 0x3c, 0x6c, 0xcc, 0xfe, 0x0c, 0x0c, 0x1e, 0x00, 0x00, 0x00, 0x00, 0x00,
00060     0x00, 0x00, 0xfe, 0xc0, 0xc0, 0xfc, 0x06, 0x06, 0x0c, 0x6c, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
00061     0x00, 0x00, 0x38, 0x60, 0x0c, 0x0c, 0xfc, 0xc6, 0x0c, 0x6c, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
00062     0x00, 0x00, 0xfe, 0xc6, 0x06, 0x0c, 0x18, 0x30, 0x30, 0x30, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00,
00063     0x00, 0x00, 0x7c, 0xc6, 0x06, 0x0c, 0x3c, 0x06, 0x0c, 0x6c, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
00064     0x00, 0x00, 0x7c, 0xc6, 0x06, 0x0c, 0x3c, 0x06, 0x0c, 0x6c, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
00065     0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00066     0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x18, 0x30, 0x00, 0x00, 0x00,
00067     0x00, 0x00, 0x00, 0x06, 0x0c, 0x18, 0x30, 0x60, 0x30, 0x18, 0x0c, 0x06, 0x00, 0x00, 0x00, 0x00,
00068     0x00, 0x00, 0x00, 0x00, 0x00, 0x7e, 0x00, 0x00, 0x00, 0x7e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00069     0x00, 0x00, 0x00, 0x60, 0x30, 0x18, 0x0c, 0x06, 0x0c, 0x18, 0x30, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00,
00070     0x00, 0x00, 0x7c, 0xc6, 0x0c, 0x18, 0x18, 0x18, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00,
00071     0x00, 0x00, 0x00, 0x7c, 0xc6, 0x0d, 0xde, 0xde, 0xde, 0xdc, 0xco, 0x7c, 0x00, 0x00, 0x00, 0x00,
00072     0x00, 0x00, 0x10, 0x38, 0x6c, 0xc6, 0xfe, 0xc6, 0x6c, 0x6c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00073     0x00, 0x00, 0xfc, 0x66, 0x66, 0x7c, 0x66, 0x66, 0x6c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00074     0x00, 0x00, 0x3c, 0x66, 0xc2, 0xc0, 0xc0, 0xc0, 0xc2, 0x66, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00,
00075     0x00, 0x00, 0xf8, 0x6c, 0x66, 0x66, 0x66, 0x66, 0x6c, 0x0f8, 0x00, 0x00, 0x00, 0x00, 0x00,
00076     0x00, 0x00, 0xfe, 0x66, 0x62, 0x68, 0x68, 0x68, 0x60, 0x62, 0x66, 0x6f, 0x00, 0x00, 0x00, 0x00, 0x00,
00077     0x00, 0x00, 0xfe, 0x66, 0x62, 0x68, 0x68, 0x68, 0x60, 0x60, 0x60, 0xf0, 0x00, 0x00, 0x00, 0x00,
00078     0x00, 0x00, 0x3c, 0x66, 0xc2, 0xc0, 0xc0, 0xde, 0xc6, 0x6c, 0x66, 0x3a, 0x00, 0x00, 0x00, 0x00, 0x00,
00079     0x00, 0x00, 0xc6, 0xc6, 0xc6, 0xc6, 0xfe, 0xc6, 0xc6, 0x0c, 0x6c, 0x00, 0x00, 0x00, 0x00, 0x00,
00080     0x00, 0x00, 0x3c, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00,
00081     0x00, 0x00, 0x1e, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0xcc, 0xcc, 0xcc, 0x78, 0x00, 0x00, 0x00, 0x00, 0x00,
00082     0x00, 0x00, 0xe6, 0x66, 0x66, 0x6c, 0x78, 0x78, 0x6c, 0x66, 0x66, 0x6e, 0x00, 0x00, 0x00, 0x00, 0x00,
00083     0x00, 0x00, 0xfc, 0x60, 0x60, 0x60, 0x60, 0x60, 0x60, 0x62, 0x66, 0x6f, 0x00, 0x00, 0x00, 0x00, 0x00,
00084     0x00, 0x00, 0x3c, 0x7e, 0xff, 0xff, 0xdb, 0x0c, 0xc3, 0xc3, 0xc3, 0xc3, 0x00, 0x00, 0x00, 0x00, 0x00,
00085     0x00, 0x00, 0xc6, 0xe6, 0xfe, 0xde, 0xc6, 0xc6, 0x6c, 0x00, 0x00, 0x00, 0x00, 0x00,
```

00086 0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,  
00087 0x00, 0x00, 0xfc, 0x66, 0x66, 0x7c, 0x60, 0x60, 0x60, 0xf0, 0x00, 0x00, 0x00, 0x00,  
00088 0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xd6, 0xde, 0x7c, 0x0c, 0xe, 0x00, 0x00,  
00089 0x00, 0x00, 0xfc, 0x66, 0x66, 0x7c, 0x6c, 0x66, 0x66, 0xe6, 0x00, 0x00, 0x00, 0x00,  
00090 0x00, 0x00, 0x7c, 0xc6, 0xc6, 0x60, 0x38, 0x0c, 0x06, 0xc6, 0xc6, 0x7c, 0x00, 0x00, 0x00,  
00091 0x00, 0x00, 0xff, 0xdb, 0x99, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00, 0x00,  
00092 0x00, 0x00, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00, 0x00,  
00093 0x00, 0x00, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0x66, 0x3c, 0x18, 0x00, 0x00,  
00094 0x00, 0x00, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0x66, 0x66, 0x66, 0x00, 0x00, 0x00,  
00095 0x00, 0x00, 0xc3, 0xc3, 0x66, 0x3c, 0x18, 0x18, 0x3c, 0x66, 0xc3, 0xc3, 0x00, 0x00, 0x00, 0x00,  
00096 0x00, 0x00, 0xc3, 0xc3, 0xc3, 0x66, 0x3c, 0x18, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00, 0x00, 0x00,  
00097 0x00, 0x00, 0xff, 0xc3, 0x86, 0x0c, 0x18, 0x30, 0x60, 0x1c, 0xc3, 0xff, 0x00, 0x00, 0x00, 0x00,  
00098 0x00, 0x00, 0x3c, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x3c, 0x00, 0x00, 0x00, 0x00,  
00099 0x00, 0x00, 0x00, 0x80, 0xc0, 0xe0, 0x70, 0x38, 0x1c, 0xe0, 0x06, 0x02, 0x00, 0x00, 0x00,  
00100 0x00, 0x00, 0x3c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x0c, 0x3c, 0x00, 0x00, 0x00, 0x00,  
00101 0x10, 0x38, 0x6c, 0xc6, 0x00,  
00102 0x00,  
00103 0x30, 0x30, 0x18, 0x00,  
00104 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x0c, 0x7c, 0xcc, 0xcc, 0x76, 0x00, 0x00, 0x00, 0x00,  
00105 0x00, 0x00, 0xe0, 0x60, 0x60, 0x78, 0x6c, 0x66, 0x66, 0x66, 0x66, 0x7c, 0x00, 0x00, 0x00, 0x00,  
00106 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6, 0xc0, 0xc0, 0xc0, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00,  
00107 0x00, 0x00, 0x1c, 0x0c, 0x0c, 0x3c, 0x6c, 0xcc, 0xcc, 0xcc, 0xcc, 0x76, 0x00, 0x00, 0x00, 0x00,  
00108 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6, 0xfe, 0xc0, 0xc0, 0x66, 0x7c, 0x00, 0x00, 0x00, 0x00,  
00109 0x00, 0x00, 0x38, 0x6c, 0x64, 0x60, 0xf0, 0x60, 0x60, 0x60, 0xf0, 0x00, 0x00, 0x00, 0x00,  
00110 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0xcc, 0xcc, 0xcc, 0xcc, 0x7c, 0x0c, 0xcc, 0x78, 0x00,  
00111 0x00, 0x00, 0xe0, 0x60, 0x60, 0x6c, 0x76, 0x66, 0x66, 0x66, 0x66, 0xe6, 0x00, 0x00, 0x00, 0x00,  
00112 0x00, 0x00, 0x18, 0x18, 0x00, 0x38, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00, 0x00, 0x00,  
00113 0x00, 0x00, 0x06, 0x06, 0x00, 0x0e, 0x06, 0x06, 0x06, 0x06, 0x06, 0x66, 0x66, 0x3c, 0x00,  
00114 0x00, 0x00, 0xe0, 0x60, 0x60, 0x66, 0x6c, 0x78, 0x6c, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x00,  
00115 0x00, 0x00, 0x38, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00, 0x00, 0x00,  
00116 0x00, 0x00, 0x00, 0x00, 0x00, 0xe6, 0xff, 0xdb, 0xdb, 0xdb, 0xdb, 0xdb, 0x00, 0x00, 0x00, 0x00,  
00117 0x00, 0x00, 0x00, 0x00, 0x00, 0xdc, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x00, 0x00, 0x00, 0x00,  
00118 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00,  
00119 0x00, 0x00, 0x00, 0x00, 0x00, 0xdc, 0x66, 0x66, 0x66, 0x66, 0x66, 0x7c, 0x60, 0x60, 0xf0, 0x00,  
00120 0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0xcc, 0xcc, 0xcc, 0xcc, 0x7c, 0x0c, 0x0c, 0x1e, 0x00,  
00121 0x00, 0x00, 0x00, 0x00, 0x00, 0xdc, 0x76, 0x66, 0x60, 0x60, 0xf0, 0x00, 0x00, 0x00, 0x00,  
00122 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0xc6, 0x60, 0x38, 0x0c, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00,  
00123 0x00, 0x00, 0x10, 0x30, 0xfc, 0x30, 0x30, 0x30, 0x36, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00,  
00124 0x00, 0x00, 0x00, 0x00, 0x00, 0xcc, 0xcc, 0xcc, 0xcc, 0x76, 0x00, 0x00, 0x00, 0x00, 0x00,  
00125 0x00, 0x00, 0x00, 0x00, 0x00, 0xc3, 0xc3, 0xc3, 0xc3, 0x66, 0x3c, 0x18, 0x00, 0x00, 0x00, 0x00,  
00126 0x00, 0x00, 0x00, 0x00, 0x00, 0xc3, 0xc3, 0xc3, 0xc3, 0xdb, 0xdb, 0xf6, 0x66, 0x00, 0x00, 0x00,  
00127 0x00, 0x00, 0x00, 0x00, 0x00, 0xc3, 0x66, 0x3c, 0x18, 0x3c, 0x66, 0xc3, 0x00, 0x00, 0x00, 0x00,  
00128 0x00, 0x00, 0x00, 0x00, 0x00, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0x7e, 0x06, 0x0c, 0xf8, 0x00,  
00129 0x00, 0x00, 0x00, 0x00, 0x00, 0xfe, 0xcc, 0x18, 0x30, 0x60, 0xc6, 0xfe, 0x00, 0x00, 0x00, 0x00,  
00130 0x00, 0x00, 0x0e, 0x18, 0x18, 0x18, 0x70, 0x18, 0x18, 0x18, 0x18, 0x18, 0x0e, 0x00, 0x00, 0x00, 0x00,  
00131 0x00, 0x00, 0x18, 0x18, 0x18, 0x18, 0x00, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00,  
00132 0x00, 0x00, 0x70, 0x18, 0x18, 0x18, 0x0e, 0x18, 0x18, 0x18, 0x18, 0x70, 0x00, 0x00, 0x00, 0x00,  
00133 0x00, 0x00, 0x76, 0xdc, 0x00,  
00134 0x00, 0x00, 0x00, 0x00, 0x10, 0x38, 0x6c, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xfe, 0x00, 0x00, 0x00, 0x00,  
00135 0x00, 0x00, 0x3c, 0x66, 0xc2, 0xc0, 0xc0, 0xc2, 0x66, 0x3c, 0x0c, 0x06, 0x7c, 0x00, 0x00, 0x00,  
00136 0x00, 0x00, 0xcc, 0x00, 0x00, 0xcc, 0xcc, 0xcc, 0xcc, 0x76, 0x00, 0x00, 0x00, 0x00, 0x00,  
00137 0x00, 0x0c, 0x18, 0x30, 0x00, 0x7c, 0xc6, 0xfe, 0xc0, 0xc0, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00,  
00138 0x00, 0x10, 0x38, 0x6c, 0x00, 0x7c, 0xc6, 0xfe, 0xc0, 0x0c, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00,  
00139 0x00, 0x00, 0xcc, 0x00, 0x00, 0x78, 0x0c, 0x7c, 0xcc, 0xcc, 0x76, 0x00, 0x00, 0x00, 0x00,  
00140 0x00, 0x60, 0x30, 0x18, 0x00, 0x78, 0x0c, 0x7c, 0xcc, 0xcc, 0x76, 0x00, 0x00, 0x00, 0x00, 0x00,  
00141 0x00, 0x38, 0x6c, 0x38, 0x00, 0x78, 0x0c, 0x7c, 0xcc, 0xcc, 0x76, 0x00, 0x00, 0x00, 0x00, 0x00,  
00142 0x00, 0x00, 0x00, 0x00, 0x00, 0x3c, 0x66, 0x60, 0x60, 0x66, 0x3c, 0x0c, 0x06, 0x3c, 0x00, 0x00, 0x00,  
00143 0x00, 0x10, 0x38, 0x6c, 0x00, 0x7c, 0xc6, 0xfe, 0xc0, 0x0c, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00,  
00144 0x00, 0x00, 0xc6, 0x00, 0x00, 0x7c, 0xc6, 0xfe, 0xc0, 0x0c, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00,  
00145 0x00, 0x60, 0x30, 0x18, 0x00, 0x7c, 0xc6, 0xfe, 0xc0, 0x0c, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00,  
00146 0x00, 0x00, 0x66, 0x00, 0x00, 0x38, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00, 0x00, 0x00,  
00147 0x00, 0x18, 0x3c, 0x66, 0x00, 0x38, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00, 0x00, 0x00,  
00148 0x00, 0x60, 0x30, 0x18, 0x00, 0x38, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00, 0x00, 0x00,  
00149 0x00, 0xc6, 0x00, 0x10, 0x38, 0x6c, 0xc6, 0xc6, 0xfe, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,  
00150 0x38, 0x6c, 0x38, 0x00, 0x38, 0x6c, 0xc6, 0xc6, 0xfe, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,  
00151 0x18, 0x30, 0x60, 0x00, 0xfe, 0x66, 0x60, 0x7c, 0x60, 0x66, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00,  
00152 0x00, 0x00, 0x00, 0x00, 0x00, 0x3b, 0x1b, 0x7e, 0xd8, 0xdc, 0x77, 0x00, 0x00, 0x00, 0x00, 0x00,  
00153 0x00, 0x00, 0x3e, 0x6c, 0xcc, 0xcc, 0xcc, 0xcc, 0xcc, 0xcc, 0xcc, 0x00, 0x00, 0x00, 0x00, 0x00,  
00154 0x00, 0x10, 0x38, 0x6c, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,  
00155 0x00, 0x00, 0xc6, 0x00, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,  
00156 0x00, 0x60, 0x30, 0x18, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,  
00157 0x00, 0x30, 0x78, 0xcc, 0x00, 0xcc, 0xcc, 0xcc, 0xcc, 0x76, 0x00, 0x00, 0x00, 0x00, 0x00,  
00158 0x00, 0x60, 0x30, 0x18, 0x00, 0xcc, 0xcc, 0xcc, 0xcc, 0xcc, 0xcc, 0x76, 0x00, 0x00, 0x00, 0x00, 0x00,  
00159 0x00, 0x00, 0xc6, 0x00, 0x00, 0x6c, 0xc6, 0xc6, 0xc6, 0xc6, 0x7e, 0x06, 0x0c, 0x78, 0x00,  
00160 0x00, 0xc6, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,  
00161 0x00, 0xc6, 0x00, 0x6c, 0xc6, 0xc6, 0xc6, 0xc6, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,  
00162 0x00, 0x18, 0x7e, 0xc3, 0x0c, 0x0c, 0x3c, 0x0e, 0x7e, 0x18, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00,  
00163 0x00, 0x38, 0x6c, 0x64, 0x60, 0xf0, 0x60, 0x60, 0x60, 0x60, 0x66, 0x0f, 0x00, 0x00, 0x00, 0x00,  
00164 0x00, 0x00, 0xc3, 0x66, 0x3c, 0x18, 0xff, 0x18, 0xff, 0x18, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00,  
00165 0x00, 0xfc, 0x66, 0x66, 0x7c, 0x62, 0x66, 0x6f, 0x66, 0x66, 0x66, 0xf3, 0x00, 0x00, 0x00, 0x00,  
00166 0x00, 0x0e, 0x1b, 0x18, 0x18, 0x7e, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0xd8, 0x70, 0x00, 0x00,  
00167 0x00, 0x18, 0x30, 0x60, 0x00, 0x78, 0x0c, 0x7c, 0xcc, 0xcc, 0x76, 0x00, 0x00, 0x00, 0x00, 0x00,  
00168 0x00, 0x0c, 0x18, 0x30, 0x00, 0x38, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00,  
00169 0x00, 0x18, 0x30, 0x60, 0x00, 0x7c, 0xc6, 0xc6, 0xc6, 0xc6, 0x6c, 0x6c, 0x7c, 0x00, 0x00, 0x00, 0x00,  
00170 0x00, 0x18, 0x30, 0x60, 0x00, 0xcc, 0xcc, 0xcc, 0xcc, 0xcc, 0x76, 0x00, 0x00, 0x00, 0x00, 0x00,  
00171 0x00, 0x00, 0x76, 0xdc, 0x00, 0xdc, 0x00, 0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x00, 0x00, 0x00, 0x00,  
00172 0x76, 0xdc, 0x00, 0xc6, 0xe6, 0xfe, 0xde, 0xce, 0xc6, 0x00, 0x00, 0x00, 0x00, 0x00,



```

00260      0x00, 0x70, 0xd8, 0x30, 0x60, 0xc8, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00261      0x00, 0x00, 0x00, 0x00, 0x7c, 0x7c, 0x7c, 0x7c, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
00262      0x00, 0x00,
00263
00264 #endif // VGA_FONT_H

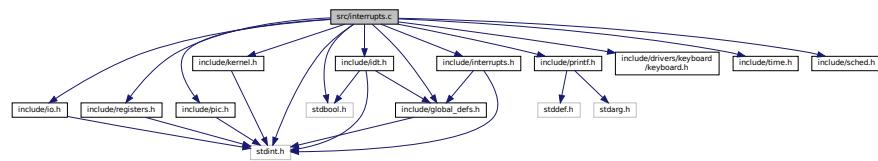
```

## 4.87 src/interrupts.c File Reference

```

#include <stdint.h>
#include "include/global_defs.h"
#include "include/idt.h"
#include "include/printf.h"
#include "include/interrupts.h"
#include "include/io.h"
#include <stdbool.h>
#include "include/registers.h"
#include "include/pic.h"
#include "include/drivers/keyboard/keyboard.h"
#include "include/time.h"
#include "include/kernel.h"
#include "include/sched.h"
Include dependency graph for interrupts.c:

```



## Functions

- void `isr_exception_handler (isr_xframe_t *frame)`
- void `irq_handler (isr_xframe_t *frame)`

## Variables

- static const char \* `exception_messages []`
- static const char \* `irq_messages []`

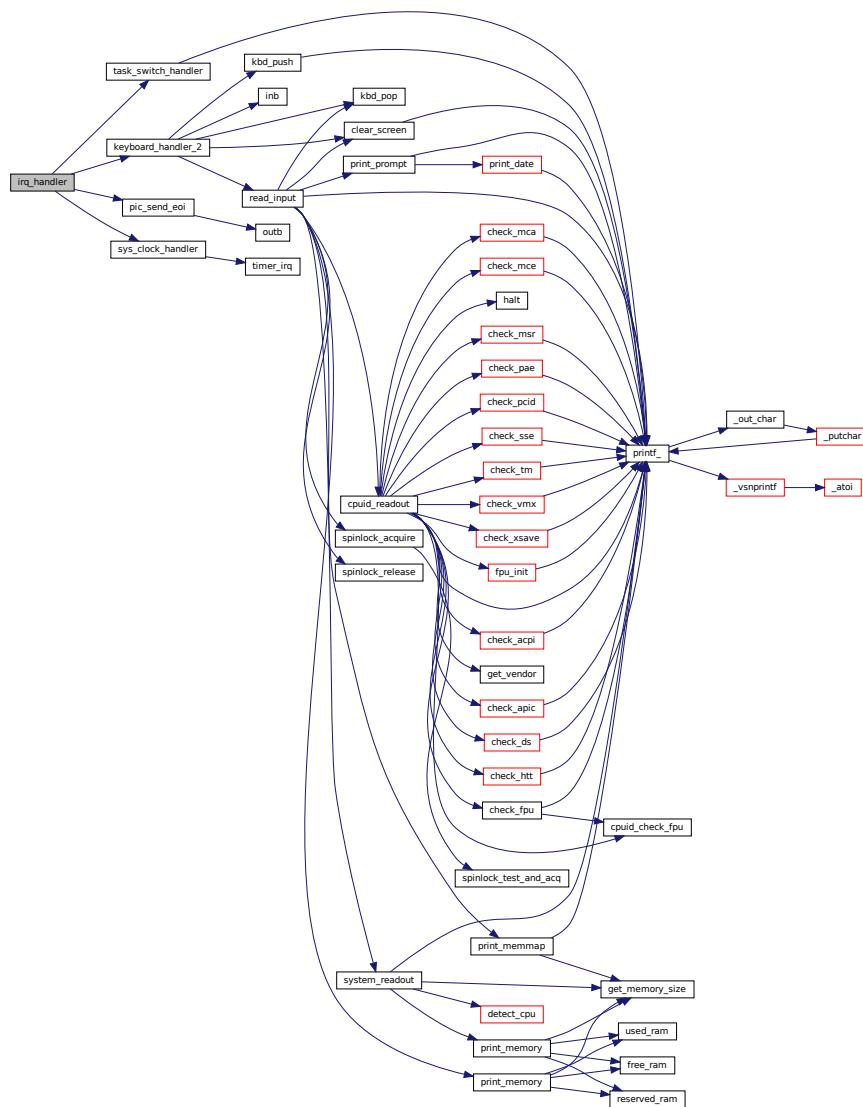
### 4.87.1 Function Documentation

### 4.87.1.1 irq\_handler()

```
void irq_handler (
    isr_xframe_t * frame )
```

Definition at line 98 of file [interrupts.c](#).

Here is the call graph for this function:

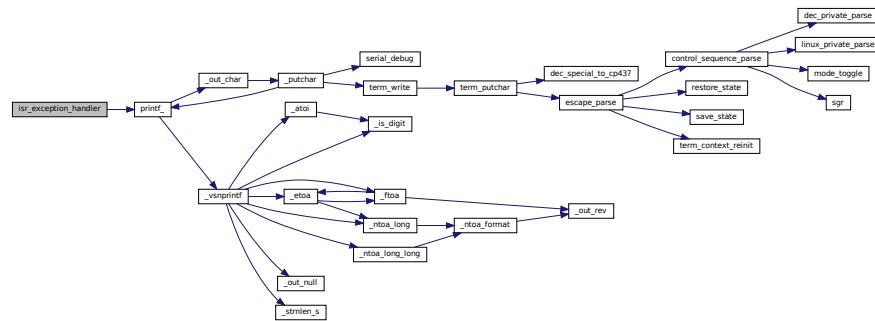


### 4.87.1.2 isr\_exception\_handler()

```
void isr_exception_handler (
    isr_xframe_t * frame )
```

Definition at line 76 of file [interrupts.c](#).

Here is the call graph for this function:



## 4.87.2 Variable Documentation

### 4.87.2.1 exception\_messages

```
const char* exception_messages[ ] [static]
```

Definition at line 15 of file [interrupts.c](#).

### 4.87.2.2 irq\_messages

```
const char* irq_messages[ ] [static]
```

#### Initial value:

```
=
{
    "System Timer",
    "Keyboard",
    "Slave PIC Link",
    "Serial Port 1",
    "Serial Port 2",
    "Reserved/Sound Card",
    "Floppy Disk Controller",
    "Parallel Port",
    "Real Time Clock",
    "Master PIC Link",
    "Reserved",
    "Reserved",
    "PS/2 Mouse",
    "Math Co-Processor",
    "Hard Disk Drive",
    "Reserved"
}
```

Definition at line 53 of file [interrupts.c](#).

## 4.88 interrupts.c

```

00001 #include <stdint.h>
00002 #include "include/global_defs.h"
00003 #include "include/idt.h"
00004 #include "include/printf.h"
00005 #include "include/interrupts.h"
00006 #include "include/io.h"
00007 #include <stdbool.h>
00008 #include "include/registers.h"
00009 #include "include/pic.h"
00010 #include "include/drivers/keyboard/keyboard.h"
00011 #include "include/time.h"
00012 #include "include/kernel.h"
00013 #include "include/sched.h"
00014
00015 static const char *exception_messages[] =
00016 {
00017     "Division By Zero",
00018     "Debug",
00019     "Non Maskable Interrupt",
00020     "Breakpoint",
00021     "Into Detected Overflow",
00022     "Out of Bounds",
00023     "Invalid Opcode",
00024     "No Coprocessor",
00025
00026     "Double Fault",
00027     "Coprocessor Segment Overrun",
00028     "Bad TSS",
00029     "Segment Not Present",
00030     "Stack Fault",
00031     "General Protection Fault",
00032     "Page Fault",
00033     "Unknown Interrupt",
00034
00035     "Coprocessor Fault",
00036     "Alignment Check",
00037     "Machine Check",
00038     "Reserved",
00039     "Reserved",
00040     "Reserved",
00041     "Reserved",
00042     "Reserved",
00043
00044     "Reserved",
00045     "Reserved",
00046     "Reserved",
00047     "Reserved",
00048     "Reserved",
00049     "Reserved",
00050     "Reserved",
00051     "Reserved"};
00052
00053 static const char *irq_messages[] =
00054 {
00055     "System Timer",
00056     "Keyboard",
00057     "Slave PIC Link",
00058     "Serial Port 1",
00059     "Serial Port 2",
00060     "Reserved/Sound Card",
00061     "Floppy Disk Controller",
00062     "Parallel Port",
00063     "Real Time Clock",
00064     "Master PIC Link",
00065     "Reserved",
00066     "Reserved",
00067     "PS/2 Mouse",
00068     "Math Co-Processor",
00069     "Hard Disk Drive",
00070     "Reserved"
00071     "Reserved"
00072 };
00073 };
00074
00075 void isr_exception_handler(isr_xframe_t *frame);
00076 void isr_exception_handler(isr_xframe_t *frame)
00077 {
00078     kerror_mode = 1;
00079
00080     printf_("%"s, "ERROR: CPU EXCEPTION: ");
00081     printf_("%"s, exception_messages[frame->base_frame.vector]);
00082     printf_("%"s, " @ ");
00083     printf_("%"s"\n", frame->base_frame.rip);
00084     printf_("%"s, "FROM VECTOR NUMBER: ");
00085

```

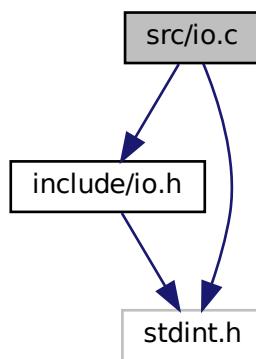
```

00086     printf_( "%i\n", frame->base_frame.vector );
00087     printf_( "%s", "ERROR CODE: " );
00088     printf_( "0x%llx\n", frame->base_frame.error_code );
00089     printf_( "%s", "CS REGISTER: " );
00090     printf_( "0x%llx\n", frame->base_frame.cs );
00091     printf_( "%s", "CR2 REGISTER: " );
00092     printf_( "0x%llx\n", frame->control_registers.cr2 );
00093
00094     __asm__ volatile("cli; hlt");
00095 }
00096
00097 void irq_handler(isr_xframe_t *frame);
00098 void irq_handler(isr_xframe_t *frame)
00099 {
00100     uint64_t vector = frame->base_frame.vector;
00102
00103     // printf_( "%s", "IRQ RECEIVED FROM: " );
00104     // printf_( "%s\n", irq_messages[vector - 32] );
00105
00106     switch (vector)
00107     {
00108
00109     case 32:
00110         sys_clock_handler();
00111         break;
00112     case 33:
00113         keyboard_handler_2();
00114         break;
00115     case 48:
00116         task_switch_handler();
00117         break;
00118     }
00119     pic_send_eoi(vector - 32);
00121 }
```

## 4.89 src/io.c File Reference

```
#include "include/io.h"
#include <stdint.h>
```

Include dependency graph for io.c:



## Functions

- `uint8_t inb (uint16_t port)`
- `void outb (uint16_t port, uint8_t val)`
- `void io_wait (void)`

## 4.89.1 Function Documentation

### 4.89.1.1 inb()

```
uint8_t inb (
    uint16_t port )
```

inb: Read a byte from an I/O port.

#### Parameters

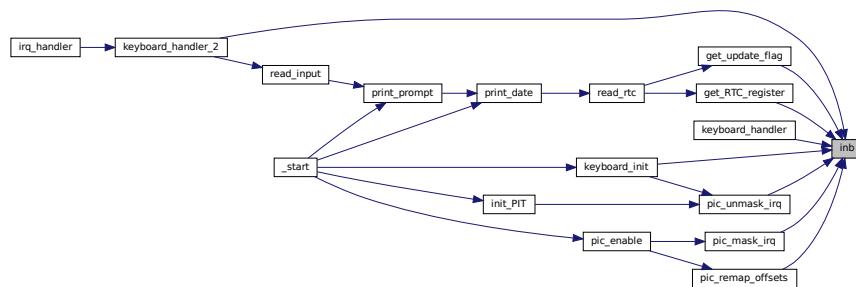
<i>port</i>	The address of the I/O port
-------------	-----------------------------

#### Returns

The read byte

Definition at line 4 of file [io.c](#).

Here is the caller graph for this function:



### 4.89.1.2 io\_wait()

```
void io_wait (
    void )
```

Definition at line 18 of file [io.c](#).

Here is the call graph for this function:



### 4.89.1.3 outb()

```
void outb (
    uint16_t port,
    uint8_t data )
```

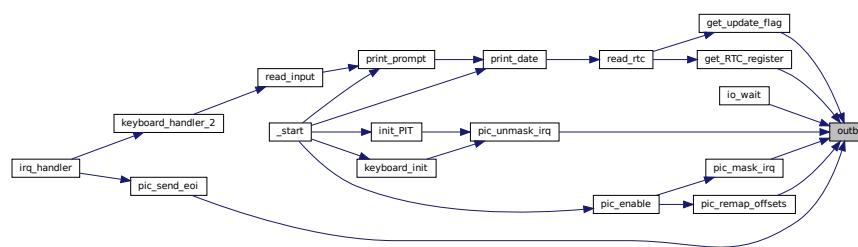
outb: Sends the given data to the given I/O port. Defined in io.s

#### Parameters

<i>port</i>	The I/O port to send the data to
<i>data</i>	The data to send to the I/O port

Definition at line 13 of file [io.c](#).

Here is the caller graph for this function:



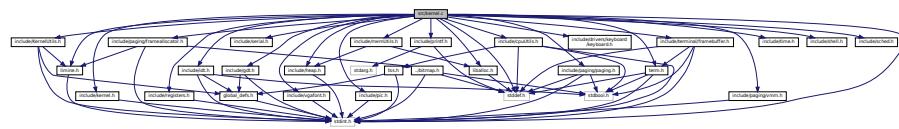
## 4.90 io.c

```
00001 #include "include/io.h"
00002 #include <stdint.h>
00003
00004 uint8_t inb(uint16_t port)
00005 {
00006     uint8_t ret;
00007     asm volatile("inb %1, %0"
00008                 : "=a"(ret)
00009                 : "Nd"(port));
00010     return ret;
00011 }
00012
00013 void outb(uint16_t port, uint8_t val)
00014 {
00015     asm volatile("outb %1, %0" ::"dN"(port), "a"(val));
00016 }
00017
00018 void io_wait(void)
00019 {
00020     outb(0x80, 0);
00021 }
```

## 4.91 src/kernel.c File Reference

```
#include <stdint.h>
#include <stddef.h>
```

```
#include "include/gdt.h"
#include "include/idt.h"
#include "include/KernelUtils.h"
#include "include/serial.h"
#include "include/limine.h"
#include "include/printf.h"
#include "include/heap.h"
#include "include/kernel.h"
#include "include/paging/frameallocator.h"
#include "include/paging/paging.h"
#include "include/registers.h"
#include "include/memUtils.h"
#include "include/vgafont.h"
#include "include/pic.h"
#include "include/drivers/keyboard/keyboard.h"
#include "include/cpuUtils.h"
#include "include/terminal/framebuffer.h"
#include "include/terminal/term.h"
#include "include/liballoc.h"
#include "include/time.h"
#include "include/shell.h"
#include "include/sched.h"
#include "include/paging/vmm.h"
Include dependency graph for kernel.c:
```



## Macros

- #define White "\033[1;00m"
- #define Red "\033[1;31m"
- #define Green "\033[1;32m"
- #define Yellow "\033[1;33m"
- #define Blue "\033[1;34m"
- #define Purple "\033[1;35m"
- #define Cyan "\033[1;36m"
- #define Black "\033[1;37m"

## Functions

- void **breakpoint** ()
 

*breakpoint()* Provides a magic breakpoint for debugging in Bochs
- void **stop\_interrupts** ()
- void **start\_interrupts** ()
- void **halt** ()
- void **task\_switch\_int** ()
- void **clear\_screen** ()
 

*Clears the screen by writing a black line to each of the characters.*
- void **print\_prompt** ()
 

*Prints the prompt to the user.*
- void **\_start** (void)
 

*\ brief Start bootloader. Called by \_init*

## Variables

- `uint32_t term_fg = 0x0055ff55`
- `uint32_t term_bg = 0x00000000`
- `KHEAPBM kheap`
- `volatile struct limine_kernel_address_request Kaddress_req`
- `volatile struct limine_terminal_request early_term`
- `uint32_t bootspace = 2`
- `uint8_t kerror_mode = 0`
- `struct term_context * term_context`
- `static struct PageTable * test_table`

### 4.91.1 Macro Definition Documentation

#### 4.91.1.1 Black

```
#define Black "\033[1;37m"
```

Definition at line 34 of file [kernel.c](#).

#### 4.91.1.2 Blue

```
#define Blue "\033[1;34m"
```

Definition at line 31 of file [kernel.c](#).

#### 4.91.1.3 Cyan

```
#define Cyan "\033[1;36m"
```

Definition at line 33 of file [kernel.c](#).

#### 4.91.1.4 Green

```
#define Green "\033[1;32m"
```

Definition at line 29 of file [kernel.c](#).

#### 4.91.1.5 Purple

```
#define Purple "\033[1;35m"
```

Definition at line [32](#) of file [kernel.c](#).

#### 4.91.1.6 Red

```
#define Red "\033[1;31m"
```

Definition at line [28](#) of file [kernel.c](#).

#### 4.91.1.7 White

```
#define White "\033[1;00m"
```

Definition at line [27](#) of file [kernel.c](#).

#### 4.91.1.8 Yellow

```
#define Yellow "\033[1;33m"
```

Definition at line [30](#) of file [kernel.c](#).

### 4.91.2 Function Documentation

#### 4.91.2.1 \_start()

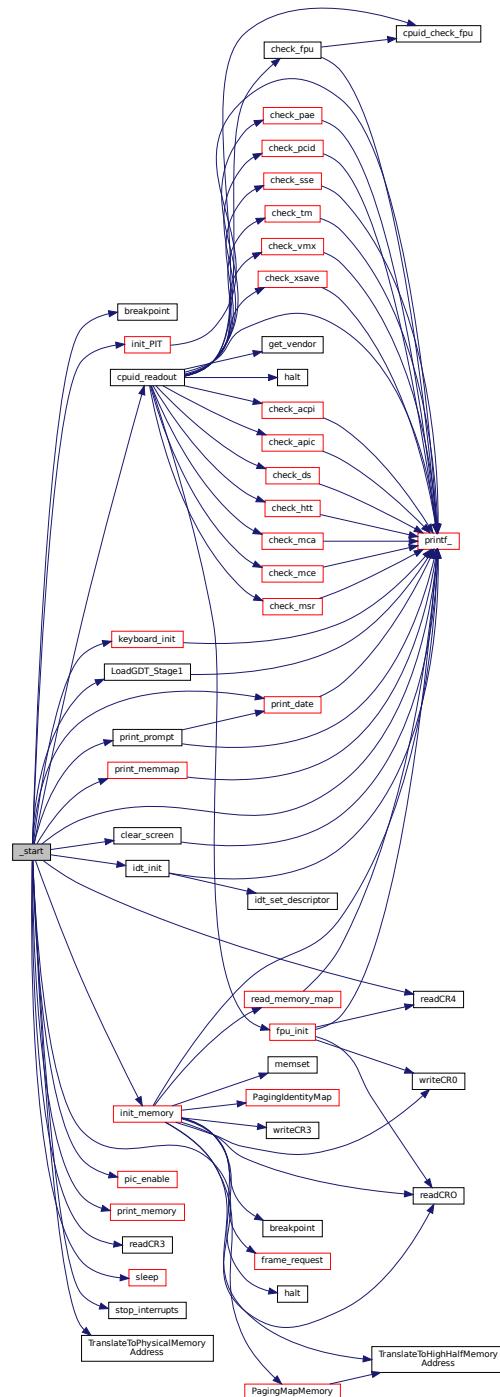
```
void _start (
    void )
```

\ brief Start bootloader. Called by \_init

print usable memory to log

Definition at line 110 of file [kernel.c](#).

Here is the call graph for this function:

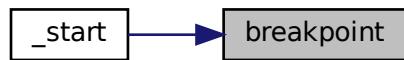


### 4.91.2.2 breakpoint()

```
void breakpoint ( )
```

**breakpoint()** Provides a magic breakpoint for debugging in Bochs

Here is the caller graph for this function:



### 4.91.2.3 clear\_screen()

```
void clear_screen ( )
```

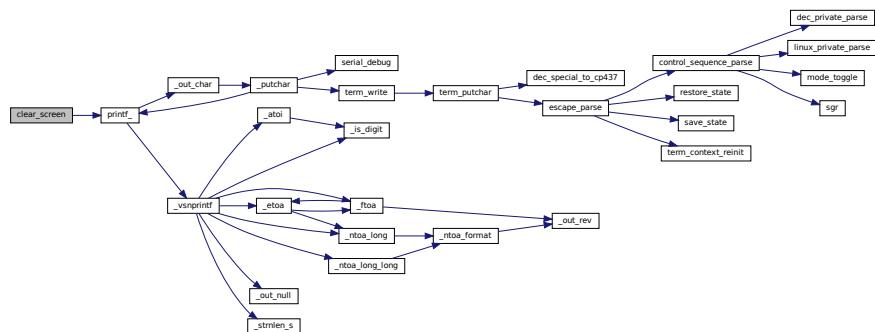
Clears the screen by writing a black line to each of the characters.

#### Returns

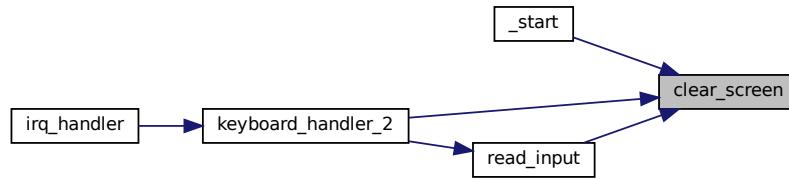
0 on success non - zero

Definition at line 80 of file [kernel.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.91.2.4 halt()

```
void halt ( )
```

#### 4.91.2.5 print\_prompt()

```
void print_prompt ( )
```

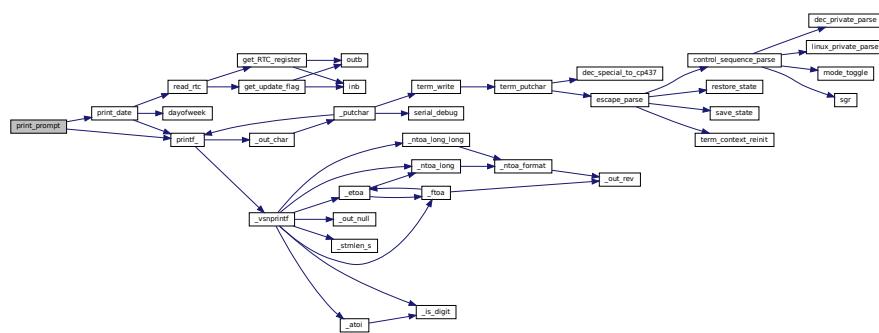
Prints the prompt to the user.

##### Returns

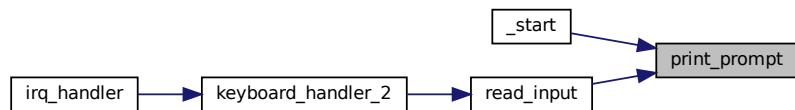
void Author : Christian Schafmeister ( 1991 ) Modifications

Definition at line 94 of file [kernel.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



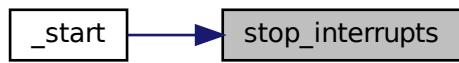
#### 4.91.2.6 start\_interrupts()

```
void start_interrupts ( )
```

#### 4.91.2.7 stop\_interrupts()

```
void stop_interrupts ( )
```

Here is the caller graph for this function:



#### 4.91.2.8 task\_switch\_int()

```
void task_switch_int ( )
```

### 4.91.3 Variable Documentation

#### 4.91.3.1 bootspace

```
uint32_t bootspace = 2
```

Definition at line 68 of file [kernel.c](#).

#### 4.91.3.2 early\_term

```
volatile struct limine_terminal_request early_term
```

##### Initial value:

```
= {  
    .id = LIMINE_TERMINAL_REQUEST,  
    .revision = 0  
}
```

Definition at line 39 of file [kernel.c](#).

#### 4.91.3.3 Kaddress\_req

```
volatile struct limine_kernel_address_request Kaddress_req
```

##### Initial value:

```
= {  
    .id = LIMINE_KERNEL_ADDRESS_REQUEST,  
    .revision = 0  
}
```

##### Attention

Limine requests can be placed anywhere, but it is important that the compiler does not optimise them away, so, usually, they should be made volatile or equivalent.

Definition at line 39 of file [kernel.c](#).

#### 4.91.3.4 kerror\_mode

```
uint8_t kerror_mode = 0
```

Definition at line 70 of file [kernel.c](#).

#### 4.91.3.5 kheap

```
KHEAPBM kheap
```

Definition at line 39 of file [kernel.c](#).

#### 4.91.3.6 term\_bg

```
uint32_t term_bg = 0x00000000
```

Definition at line 37 of file [kernel.c](#).

#### 4.91.3.7 term\_context

```
struct term_context* term_context
```

Definition at line 72 of file [kernel.c](#).

#### 4.91.3.8 term\_fg

```
uint32_t term_fg = 0x0055ff55
```

Definition at line 36 of file [kernel.c](#).

#### 4.91.3.9 test\_table

```
struct PageTable* test_table [static]
```

Definition at line 74 of file [kernel.c](#).

## 4.92 kernel.c

```
00001 #include <stdint.h>
00002 #include <stddef.h>
00003 #include "include/gdt.h"
00004 #include "include/idt.h"
00005 #include "include/KernelUtils.h"
00006 #include "include/serial.h"
00007 #include "include/limine.h"
00008 #include "include/printf.h"
00009 #include "include/heap.h"
00010 #include "include/kernel.h"
00011 #include "include/paging/frameallocator.h"
00012 #include "include/paging/paging.h"
00013 #include "include/registers.h"
00014 #include "include/memUtils.h"
00015 #include "include/vgafont.h"
00016 #include "include/pic.h"
00017 #include "include/drivers/keyboard/keyboard.h"
00018 #include "include/cpuUtils.h"
00019 #include "include/terminal/framebuffer.h"
00020 #include "include/terminal/term.h"
00021 #include "include/liballoc.h"
00022 #include "include/time.h"
00023 #include "include/shell.h"
00024 #include "include/sched.h"
00025 #include "include/paging/vmm.h"
00026
00027 #define White "\033[1;00m"
00028 #define Red "\033[1;31m"
00029 #define Green "\033[1;32m"
```

```

00030 #define Yellow "\033[1;33m"
00031 #define Blue "\033[1;34m"
00032 #define Purple "\033[1;35m"
00033 #define Cyan "\033[1;36m"
00034 #define Black "\033[1;37m"
00035
00036 uint32_t term_fg = 0x0055ffff;
00037 uint32_t term_bg = 0x00000000;
00038
00039 KHEAPBM kheap;
00040
00044
00045 volatile struct limine_kernel_address_request Kaddress_req = {
00046
00047     .id = LIMINE_KERNEL_ADDRESS_REQUEST,
00048     .revision = 0
00049
00050 };
00051
00052 volatile struct limine_terminal_request early_term = {
00053
00054     .id = LIMINE_TERMINAL_REQUEST,
00055     .revision = 0
00056
00057 };
00058
00059 extern void breakpoint();
00060 extern void stop_interrupts();
00061 extern void start_interrupts();
00062 extern void halt();
00063 extern void task_switch_int();
00064
00065
00066 uint32_t bootspace = 2;
00067
00068 uint8_t kerror_mode = 0;
00069
00070 struct term_context *term_context;
00071
00072 static struct PageTable *test_table;
00073
00074 void clear_screen()
00075 {
00076
00077     for (uint64_t i = 0; i < 500; i++)
00078     {
00079
00080         printf_("%s\n", "");
00081     }
00082
00083
00084
00085
00086
00087
00088
00089
00090
00091
00092
00093
00094 void print_prompt()
00095 {
00096
00097     print_date();
00098     printf_("%"s\n", "Please Select an Option.");
00099     printf_("%"s\n", "1) Memory Map");
00100    printf_("%"s\n", "2) CPUID Feature Readout");
00101    printf_("%"s\n", "3) RAM Readout");
00102    printf_("%"s\n", "4) System Readout");
00103    printf_("%"s\n", "Please Use The Power Button to Shutdown.");
00104 }
00105
00106
00107
00108 void _start(void)
00109 {
00110
00111     if (early_term.response == NULL || early_term.response->terminal_count < 1)
00112     {
00113
00114         bootspace = 1;
00115
00116         printf_("%s\n", "Bootloader Terminal Offline Using Serial Only!");
00117
00118     }
00119
00120
00121     printf_("%"s", "Early Terminal Using Framebuffer At Physical Address: ");
00122     printf_("0x%llx\n",
00123             TranslateToPhysicalMemoryAddress(early_term.response->terminals[0]->framebuffer));
00124     printf_("%"s", "And At Virtual Address: ");
00125     printf_("0x%llx\n", early_term.response->terminals[0]->framebuffer);
00126
00127     print_date();
00128     cpuid_readout();
00129
00130     breakpoint();
00131
00132     stop_interrupts();

```

```

00133
00134     LoadGDT_Stage1();
00135
00136     printf_(""%s\n", "Loaded GDT");
00137
00138     breakpoint();
00139
00140     idt_init();
00141
00142     printf_(""%s\n", "Loaded IDT");
00143
00144     pic_enable();
00145
00146     printf_(""%s\n", "PICs Online");
00147
00148     print_memmap();
00149
00150     // @brief Kernel Addresses
00151     if (Kaddress_req.response == NULL)
00152     {
00153         printf_(""%s\n", "!!!Error While Fetching Kernel Addresses!!!");
00154     }
00155     else
00156     {
00157         printf_(""%s\n", "Kernel Base Addresses Are As Follows:");
00158         printf_(""%s", "Physical Address: ");
00159         printf_(""0x%llx\n", Kaddress_req.response->physical_base);
00160         printf_(""%s", "Virtual Address: ");
00161         printf_(""0x%llx\n", Kaddress_req.response->virtual_base);
00162         printf_(""%s\n", "-----");
00163     }
00164
00165     init_PIT();
00166
00167     breakpoint();
00168
00169     // read_memory_map();
00170
00171     print_memory();
00172
00173     init_memory();
00174
00175
00176     printf_(""%s", "CR3: ");
00177     printf_(""0x%llx\n", readCR3());
00178     printf_(""%s", "CRO: ");
00179     printf_(""0x%llx\n", readCRO());
00180     printf_(""%s", "CR4: ");
00181     printf_(""0x%llx\n", readCR4());
00182
00183     printf_(""%s\n", "Handing Control to Standalone Terminal...");
00184
00185     bootspace = 3;
00186
00187     early_term.response->write(early_term.response->terminals[0], NULL, LIMINE_TERMINAL_FULL_REFRESH);
00188
00189     /*      bootspace = 1;
00190
00191         term_context = fbterm_init(malloc, fbr_req.response->framebuffers[0]->address,
00192                                     fbr_req.response->framebuffers[0]->width, fbr_req.response->framebuffers[0]->height,
00193                                     fbr_req.response->framebuffers[0]->pitch, NULL, NULL, NULL, NULL,
00194                                     &term_bg, &term_fg, NULL, 0, 0, 0,
00195                                     1, 1, 1);
00196
00197
00198         bootspace = 0; */
00199
00200     keyboard_init();
00201
00202     clear_screen();
00203
00204     print_memory();
00205
00206     printf_(""%s\n", "Kernel Loaded");
00207
00208     printf_(""%s", "Loadtime roughly: ");
00209
00210     printf_(""%i", system_timer_ms);
00211
00212     printf_(""%s", ".");
00213
00214     printf_(""%i", system_timer_fractions);
00215
00216     printf_(""%s\n", " ms.");
00217
00218

```

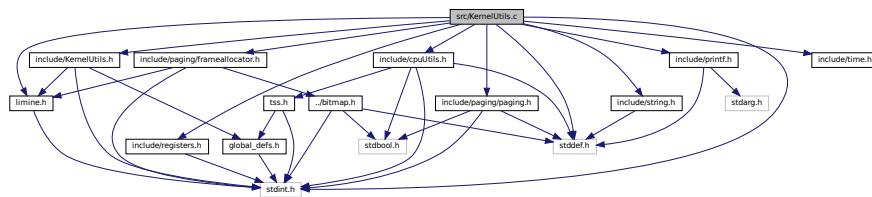
```

00219     print_date();
00220
00221     printf_( "%s\n", "Powered by VoyagerOS64 V0.0.5 (Diagnostic Branch)" );
00222
00223     sleep(100);
00224
00225     clear_screen();
00226
00227     print_prompt();
00228
00229     // cpuid_readout();
00230
00231     // init_multitasking();
00232
00233     // Just chill until needed
00234     while (1)
00235     {
00236         // read_input();
00237     }
00238 }
```

## 4.93 src/KernelUtils.c File Reference

```

#include "include/KernelUtils.h"
#include <stddef.h>
#include <stdint.h>
#include "include/limine.h"
#include "include/paging/frameallocator.h"
#include "include/paging/paging.h"
#include "include/string.h"
#include "include/registers.h"
#include "include/time.h"
#include "include/cpuUtils.h"
#include "include/printf.h"
Include dependency graph for KernelUtils.c:
```



## Macros

- #define ALIGN\_DOWN(value, align) ((value / align) \* align)
- #define ALIGN\_UP(value, align) (((value + (align - 1)) / align) \* align)

## Typedefs

- typedef char symbol[]

## Functions

- void `breakpoint()`
- void `halt()`
- void `system_readout()`
- `uint64_t get_memory_size()`
- void `print_memmap()`
- void `init_memory()`
- void `print_memory()`

## Variables

- `uint8_t * frameBitmap`
- `symbol text_start_addr`
- `symbol text_end_addr`
- `symbol rodata_start_addr`
- `symbol rodata_end_addr`
- `symbol data_start_addr`
- `symbol data_end_addr`
- `volatile struct limine_memmap_request memmap_req`
- `volatile struct limine_framebuffer_request fbr_req`
- `volatile struct limine_smp_request smp_req`
- `const struct kswitches k_mode`
- `volatile struct limine_kernel_address_request Kaddress_req`
- `static struct PageTable * page_table`

### 4.93.1 Macro Definition Documentation

#### 4.93.1.1 ALIGN\_DOWN

```
#define ALIGN_DOWN(  
    value,  
    align) ((value / align) * align)
```

#### 4.93.1.2 ALIGN\_UP

```
#define ALIGN_UP(  
    value,  
    align) (((value + (align - 1)) / align) * align)
```

### 4.93.2 Typedef Documentation

#### 4.93.2.1 symbol

```
typedef char symbol[]
```

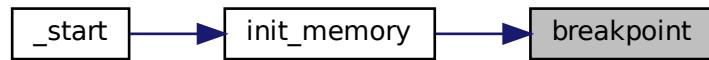
Definition at line 17 of file [KernelUtils.c](#).

### 4.93.3 Function Documentation

#### 4.93.3.1 breakpoint()

```
void breakpoint( )
```

Here is the caller graph for this function:

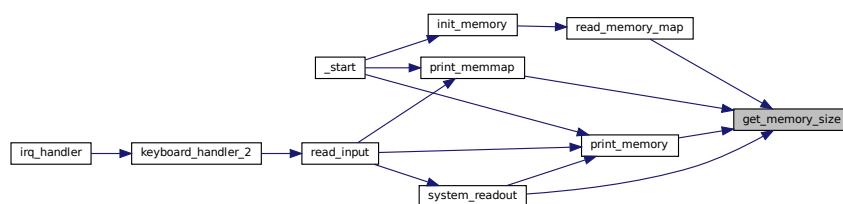


#### 4.93.3.2 get\_memory\_size()

```
uint64_t get_memory_size( )
```

Definition at line 72 of file [KernelUtils.c](#).

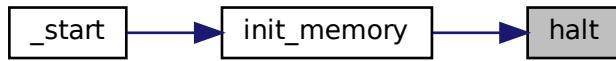
Here is the caller graph for this function:



### 4.93.3.3 halt()

```
void halt ( )
```

Here is the caller graph for this function:

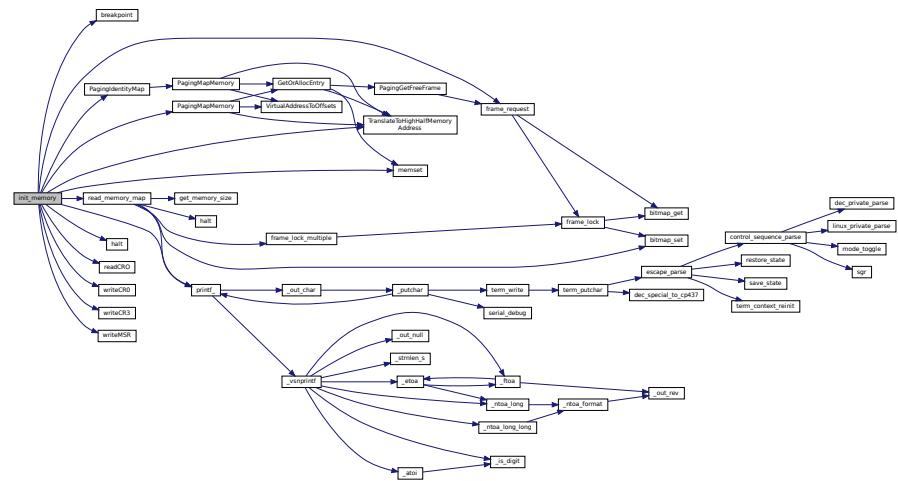


#### 4.93.3.4 init\_memory()

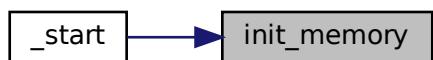
```
void init_memory ( )
```

Definition at line 185 of file KernelUtils.c.

Here is the call graph for this function:



Here is the caller graph for this function:

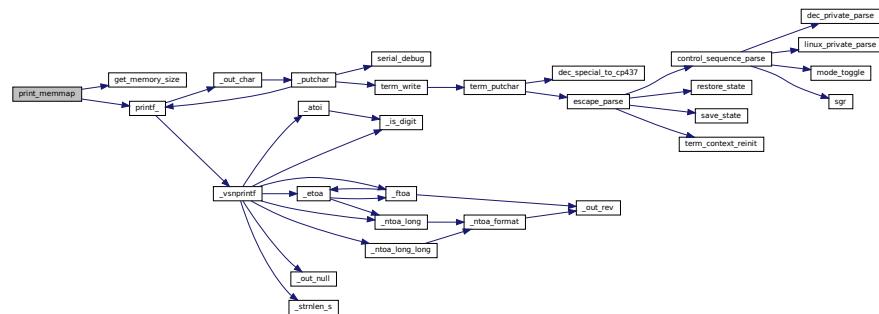


#### 4.93.3.5 print\_memmap()

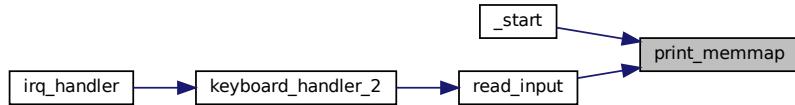
```
void print_memmap ( )
```

Definition at line 89 of file [KernelUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

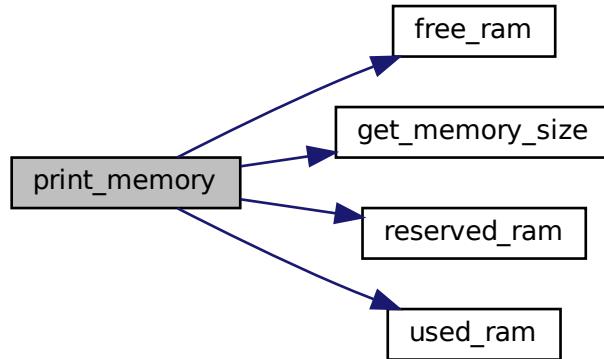


#### 4.93.3.6 print\_memory()

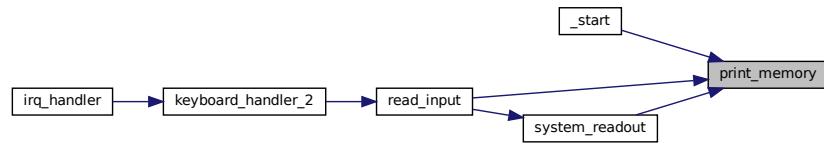
```
void print_memory ( )
```

Definition at line 310 of file [KernelUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

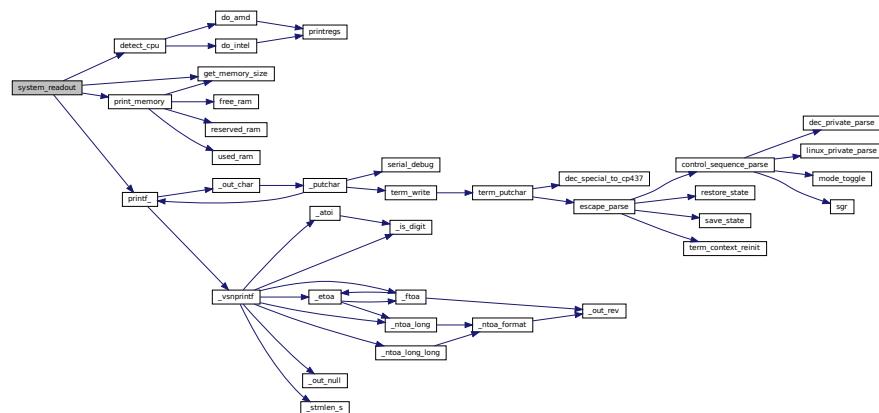


#### 4.93.3.7 system\_readout()

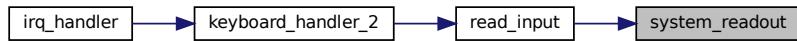
```
void system_readout( )
```

Definition at line 49 of file [KernelUtils.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.93.4 Variable Documentation

##### 4.93.4.1 data\_end\_addr

`symbol data_end_addr`

Definition at line 21 of file [KernelUtils.c](#).

##### 4.93.4.2 data\_start\_addr

`symbol data_start_addr`

Definition at line 21 of file [KernelUtils.c](#).

##### 4.93.4.3 fbr\_req

`volatile struct limine_framebuffer_request fbr_req`

###### Initial value:

```
= {  
    .id = LIMINE_FRAMEBUFFER_REQUEST,  
    .revision = 0  
}
```

Definition at line 21 of file [KernelUtils.c](#).

##### 4.93.4.4 frameBitmap

`uint8_t* frameBitmap [extern]`

Definition at line 13 of file [frameallocator.c](#).

#### 4.93.4.5 k\_mode

```
const struct kswitches k_mode
```

**Initial value:**

```
= {  
    .mem_readout_unit = 0  
}
```

Definition at line 21 of file [KernelUtils.c](#).

#### 4.93.4.6 Kaddress\_req

```
volatile struct limine_kernel_address_request Kaddress_req [extern]
```

**Attention**

Limine requests can be placed anywhere, but it is important that the compiler does not optimise them away, so, usually, they should be made volatile or equivalent.

Definition at line 39 of file [kernel.c](#).

#### 4.93.4.7 memmap\_req

```
volatile struct limine_memmap_request memmap_req
```

**Initial value:**

```
= {  
    .id = LIMINE_MEMMAP_REQUEST,  
    .revision = 0}
```

Definition at line 21 of file [KernelUtils.c](#).

#### 4.93.4.8 page\_table

```
struct PageTable* page_table [static]
```

Definition at line 183 of file [KernelUtils.c](#).

#### 4.93.4.9 rodata\_end\_addr

```
symbol rodata_end_addr
```

Definition at line 20 of file [KernelUtils.c](#).

#### 4.93.4.10 rodata\_start\_addr

```
symbol rodata_start_addr
```

Definition at line 20 of file [KernelUtils.c](#).

#### 4.93.4.11 smp\_req

```
volatile struct limine_smp_request smp_req
```

##### Initial value:

```
= {
    .id = LIMINE_SMP_REQUEST,
    .revision = 0
}
```

Definition at line 21 of file [KernelUtils.c](#).

#### 4.93.4.12 text\_end\_addr

```
symbol text_end_addr
```

Definition at line 19 of file [KernelUtils.c](#).

#### 4.93.4.13 text\_start\_addr

```
symbol text_start_addr [extern]
```

### 4.94 KernelUtils.c

```
00001 #include "include/KernelUtils.h"
00002 #include <stddef.h>
00003 #include <stdint.h>
00004 #include "include/limine.h"
00005 #include "include/paging/frameallocator.h"
00006 #include "include/paging/paging.h"
00007 #include "include/string.h"
00008 #include "include/registers.h"
00009 #include "include/time.h"
00010 #include "include/cpuUtils.h"
00011 #include "include/printf.h"
00012
00013 extern void breakpoint();
00014 extern uint8_t *frameBitmap;
00015 extern void halt();
00016
00017 typedef char symbol[];
00018
00019 extern symbol text_start_addr, text_end_addr,
00020     rodata_start_addr, rodata_end_addr,
00021     data_start_addr, data_end_addr;
00022
00023 volatile struct limine_memmap_request memmap_req = {
00024     .id = LIMINE_MEMMAP_REQUEST,
00025     .revision = 0};
```

```

00026
00027 volatile struct limine_framebuffer_request fbr_req = {
00028
00029     .id = LIMINE_FRAMEBUFFER_REQUEST,
00030     .revision = 0
00031
00032 };
00033
00034 volatile struct limine_smp_request smp_req = {
00035
00036     .id = LIMINE_SMP_REQUEST,
00037     .revision = 0
00038
00039 };
00040
00041 const struct kswitches k_mode = {
00042
00043     .mem_readout_unit = 0
00044
00045 };
00046
00047 extern volatile struct limine_kernel_address_request Kaddress_req;
00048
00049 void system_readout()
00050 {
00051
00052     uint64_t memory_size = get_memory_size() / 1000;
00053
00054     uint8_t core_count = smp_req.response->cpu_count;
00055
00056     printf_("%"s\n", "-----");
00057     printf_("%"s\n", "| System Overview |");
00058     printf_("%"s\n", "-----");
00059     print_memory();
00060     printf_("%"s\n", "-----");
00061
00062     printf_("Number of Physical Cores: %lli\n", core_count);
00063
00064     printf_("%"s\n", "-----");
00065     printf_("%"s\n", "NOTE: Database is Incomplete And May Not be Accurate as a Result!");
00066
00067     detect_cpu();
00068
00069     printf_("%"s\n", "-----");
00070 }
00071
00072 uint64_t get_memory_size()
00073 {
00074     static uint64_t memorySize = 0;
00075
00076     if (memorySize > 0)
00077     {
00078         return memorySize;
00079     }
00080
00081     for (uint64_t i = 0; i < memmap_req.response->entry_count; i++)
00082     {
00083         memorySize += memmap_req.response->entries[i]->length;
00084     }
00085
00086     return memorySize;
00087 }
00088
00089 void print_memmap()
00090 {
00091     int size = memmap_req.response->entry_count;
00092
00093     int num_useable = 0;
00094     int num_bad = 0;
00095     int num_reclaim_b1 = 0;
00096     int num_reclaim_acpi = 0;
00097
00098     printf_("%"s\n", "-----");
00099     printf_("%"s\n", "| MEMORY MAP |");
00100     printf_("%"s\n", "-----");
00101     printf_("%"s\n", "Type Legend: ");
00102
00103     printf_("%"s", "Usable: ");
00104     printf_("%"i"\n", 0);
00105     printf_("%"s", "Reserved: ");
00106     printf_("%"i"\n", 1);
00107     printf_("%"s", "ACPI Reclaimable: ");
00108     printf_("%"i"\n", 2);
00109     printf_("%"s", "ACPI NVS: ");
00110     printf_("%"i"\n", 3);
00111     printf_("%"s", "Bad Memory: ");
00112     printf_("%"i"\n", 4);

```

```

00113     printf_("\"%s\", \"Bootloader Reclaimable: \"");
00114     printf_("\"%i\\n\", 5);
00115     printf_("\"%s\", \"Kernel And Modules: \"");
00116     printf_("\"%i\\n\", 6);
00117     printf_("\"%s\", \"Framebuffer: \"");
00118     printf_("\"%i\\n\", 7);
00119     printf_("\"%s\\n\", \"-----");
00120     printf_("\"%s\", \"Number Of Entries: \"");
00121     printf_("\"%i\\n\", size);
00122     printf_("\"%s\\n\", \"-----");
00123
00124     printf_("\"%s\\n\", \"Memory Map Is As Follows: \"");
00125
00126     for (size_t i = 0; i < size; ++i)
00127     {
00128
00129         printf_("\"%s\", \"Info For Entry Number: \"");
00130         printf_("\"%i\\n\", i + 1);
00131         printf_("\"%s\", \"Entry Base: \"");
00132         printf_("\"0x%llx\\n\", memmap_req.response->entries[i]->base");
00133         printf_("\"%s\", \"Entry Limit: \"");
00134         printf_("\"0x%llx\\n\", memmap_req.response->entries[i]->length");
00135         printf_("\"%s\", \"Entry Type: \"");
00136         printf_("\"%i\\n\", memmap_req.response->entries[i]->type");
00137         printf_("\"%s\\n\", \"-----");
00138
00139         if (pit_armed == 1)
00140         {
00141
00142             // sleep(50);
00143         }
00144
00145         if (memmap_req.response->entries[i]->type == 0)
00146         {
00147
00148             num_useable++;
00149         }
00150
00151         if (memmap_req.response->entries[i]->type == 4)
00152         {
00153
00154             num_bad++;
00155         }
00156
00157         if (memmap_req.response->entries[i]->type == 5)
00158         {
00159
00160             num_reclaim_b1++;
00161         }
00162
00163         if (memmap_req.response->entries[i]->type == 2)
00164         {
00165
00166             num_reclaim_acpi++;
00167         }
00168     }
00169
00170     printf_("\"%s\", \"Number of Usable Entries: \"");
00171     printf_("\"%i\\n\", num_useable);
00172     printf_("\"%s\", \"Number of Bad Entries: \"");
00173     printf_("\"%i\\n\", num_bad);
00174     printf_("\"%s\", \"Number of Bootloader Reclaimable Entries: \"");
00175     printf_("\"%i\\n\", num_reclaim_b1);
00176     printf_("\"%s\", \"Number of ACPI Reclaimable Entries: \"");
00177     printf_("\"%i\\n\", num_reclaim_acpi);
00178     printf_("\"%s\", \"Memory Size: \"");
00179     printf_("\"0x%llx\\n\", get_memory_size());
00180     printf_("\"%s\\n\", \"-----");
00181 }
00182
00183 static struct PageTable *page_table;
00184
00185 void init_memory()
00186 {
00187     read_memory_map();
00188
00189     page_table = (struct PageTable *)frame_request();
00190
00191     memset(page_table, 0, sizeof(struct PageTable));
00192
00193     printf_("\"%s\\n\", \"Initializing Paging\"");
00194
00195     breakpoint();
00196
00197     printf_("\"%s\\n\", \"Preallocating Upper Region\"");
00198
00199     for (uint64_t i = 256; i < 512; i++)

```

```

00200     {
00201         void *page = frame_request();
00202
00203         if (page == 0x0)
00204         {
00205
00206             printf_( "%s\n", "!!!Kernel Panic!!!");
00207             printf_( "%s", "Attempt to preallocate page with invalid address at table index: ");
00208             printf_( "%i\n", i );
00209             printf_( "%s", "Requested address: " );
00210             printf_( "0x%llx\n", page );
00211             printf_( "%s\n", "!!!Kernel Panic!!!");
00212             halt();
00213
00214         }
00215
00216         memset(page, 0, 0x1000);
00217
00218         page_table->entries[i] = (uint64_t)page | PAGING_FLAG_PRESENT | PAGING_FLAG_WRITABLE;
00219     }
00220
00221     // Enable Write Protection
00222     writeCR0(readCR0() | (1 << 16));
00223
00224     // Program the PAT
00225     writeMSR(0x0277, 0x000000005010406);
00226
00227     printf_( "%s\n", "Mapping Memory Map");
00228
00229 #define ALIGN_DOWN(value, align) ((value / align) * align)
00230 #define ALIGN_UP(value, align) (((value + (align - 1)) / align) * align)
00231
00232     uint64_t textStart = ALIGN_DOWN((uint64_t)text_start_addr, 0x1000);
00233     uint64_t textEnd = ALIGN_UP((uint64_t)text_end_addr, 0x1000);
00234     uint64_t rodataStart = ALIGN_DOWN((uint64_t)rodata_start_addr, 0x1000);
00235     uint64_t rodataEnd = ALIGN_UP((uint64_t)rodata_end_addr, 0x1000);
00236     uint64_t dataStart = ALIGN_DOWN((uint64_t)data_start_addr, 0x1000);
00237     uint64_t dataEnd = ALIGN_UP((uint64_t)data_end_addr, 0x1000);
00238
00239     printf_( "map text\n");
00240
00241     for (uint64_t textAddress = textStart; textAddress < textEnd; textAddress += 0x1000)
00242     {
00243         uint64_t target = textAddress - Kaddress_req.response->virtual_base +
00244         Kaddress_req.response->physical_base;
00245
00246         PagingMapMemory(page_table, (void *)textAddress, (void *)target, PAGING_FLAG_PRESENT);
00247     }
00248
00249     printf_( "map rodata\n");
00250
00251     for (uint64_t rodataAddress = rodataStart; rodataAddress < rodataEnd; rodataAddress += 0x1000)
00252     {
00253         uint64_t target = rodataAddress - Kaddress_req.response->virtual_base +
00254         Kaddress_req.response->physical_base;
00255
00256         PagingMapMemory(page_table, (void *)rodataAddress, (void *)target, PAGING_FLAG_PRESENT |
00257         PAGING_FLAG_NO_EXECUTE);
00258
00259     printf_( "map data\n");
00260
00261     for (uint64_t dataAddress = dataStart; dataAddress < dataEnd; dataAddress += 0x1000)
00262     {
00263         uint64_t target = dataAddress - Kaddress_req.response->virtual_base +
00264         Kaddress_req.response->physical_base;
00265
00266         PagingMapMemory(page_table, (void *)dataAddress, (void *)target, PAGING_FLAG_PRESENT |
00267         PAGING_FLAG_WRITABLE | PAGING_FLAG_NO_EXECUTE);
00268     }
00269
00270     printf_( "map global memory\n");
00271
00272     for (uintptr_t addr = 0x1000; addr < 0x100000000; addr += 0x1000)
00273     {
00274         PagingIdentityMap(page_table, addr, PAGING_FLAG_PRESENT | PAGING_FLAG_WRITABLE);
00275         PagingMapMemory(page_table, TranslateToHighHalfMemoryAddress(addr), addr, PAGING_FLAG_PRESENT |
00276         PAGING_FLAG_WRITABLE | PAGING_FLAG_NO_EXECUTE);
00277     }
00278
00279     printf_( "map kernel and modules\n");
00280
00281     for (uint64_t i = 0; i < memmap_req.response->entry_count; i++)
00282     {
00283         struct limine_memmap_entry *entry = memmap_req.response->entries[i];
00284
00285         uint64_t base = ALIGN_DOWN(entry->base, 0x1000);

```

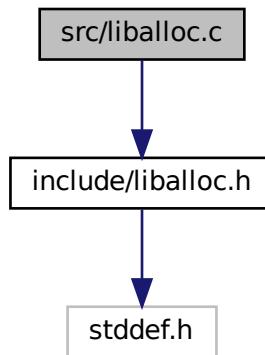
```

00281     uint64_t top = ALIGN_UP(entry->base + entry->length, 0x1000);
00282
00283     if (top <= 0x100000000)
00284     {
00285         continue;
00286     }
00287
00288     for (uint64_t j = base; j < top; j += 0x1000)
00289     {
00290
00291         if (j < 0x100000000)
00292         {
00293             continue;
00294         }
00295
00296         PagingIdentityMap(page_table, j, PAGING_FLAG_PRESENT | PAGING_FLAG_WRITABLE);
00297         PagingMapMemory(page_table, TranslateToHighHalfMemoryAddress(j), j, PAGING_FLAG_PRESENT |
00298 PAGING_FLAG_WRITABLE | PAGING_FLAG_NO_EXECUTE);
00299     }
00300
00301     frameBitmap = (uint8_t *)TranslateToHighHalfMemoryAddress(frameBitmap);
00302
00303     writeCR3((uint64_t)page_table);
00304
00305     printf_("Wrote CR3\n");
00306
00307     breakpoint();
00308 }
00309
00310 void print_memory()
00311 {
00312     double exp;
00313
00314     switch (k_mode.mem_readout_unit)
00315     {
00316
00317     case 0:
00318         exp = 1073741824;
00319
00320         printf("Total Memory: %.2f Gb.\n", (double)get_memory_size() / exp);
00321         printf("Free Memory: %.2f Gb.\n", (double)free_ram() / exp);
00322         printf("Used Memory: %.2f Gb.\n", (double)used_ram() / exp);
00323         printf("Reserved Memory: %.2f Gb.\n", (double)reserved_ram() / exp);
00324         break;
00325
00326     case 1:
00327         exp = 1048576;
00328
00329         printf("Total Memory: %.2f Mb.\n", (double)get_memory_size() / exp);
00330         printf("Free Memory: %.2f Mb.\n", (double)free_ram() / exp);
00331         printf("Used Memory: %.2f Mb.\n", (double)used_ram() / exp);
00332         printf("Reserved Memory: %.2f Mb.\n", (double)reserved_ram() / exp);
00333         break;
00334     case 2:
00335         exp = 1024;
00336
00337         printf("Total Memory: %.2f Kb.\n", (double)get_memory_size() / exp);
00338         printf("Free Memory: %.2f Kb.\n", (double)free_ram() / exp);
00339         printf("Used Memory: %.2f Kb.\n", (double)used_ram() / exp);
00340         printf("Reserved Memory: %.2f Kb.\n", (double)reserved_ram() / exp);
00341         break;
00342     case 3:
00343         exp = 1;
00344
00345         printf("Total Memory: %.2f Bytes.\n", (double)get_memory_size() / exp);
00346         printf("Free Memory: %.2f Bytes.\n", (double)free_ram() / exp);
00347         printf("Used Memory: %.2f Bytes.\n", (double)used_ram() / exp);
00348         printf("Reserved Memory: %.2f Bytes.\n", (double)reserved_ram() / exp);
00349         break;
00350     default:
00351         printf("%s\n", "ERROR: Invalid value in k_mode.mem_readout_unit switch!");
00352         break;
00353     }
00354 }
```

## 4.95 src/liballoc.c File Reference

```
#include "include/liballoc.h"
```

Include dependency graph for liballoc.c:



## Macros

- #define LIBALLOC\_MAGIC 0xc001c0de
- #define MAXCOMPLETE 5
- #define MAXEXP 32
- #define MINEXP 8
- #define MODE\_BEST 0
- #define MODE\_INSTANT 1
- #define MODE MODE\_BEST

## Functions

- static int `getexp` (unsigned int size)
- static void \* `liballoc_memset` (void \*s, int c, size\_t n)
- static void \* `liballoc_memcpy` (void \*s1, const void \*s2, size\_t n)
- static void `insert_tag` (struct `boundary_tag` \*tag, int index)
- static void `remove_tag` (struct `boundary_tag` \*tag)
- static struct `boundary_tag` \* `melt_left` (struct `boundary_tag` \*tag)
- static struct `boundary_tag` \* `absorb_right` (struct `boundary_tag` \*tag)
- static struct `boundary_tag` \* `split_tag` (struct `boundary_tag` \*tag)
- static struct `boundary_tag` \* `allocate_new_tag` (unsigned int size)
- void \* `malloc` (size\_t size)
- void `free` (void \*ptr)
- void \* `calloc` (size\_t nobj, size\_t size)
- void \* `realloc` (void \*p, size\_t size)

## Variables

- struct `boundary_tag` \* `_freePages` [MAXEXP]
- int `_completePages` [MAXEXP]
- static int `_initialized` = 0
- static int `_pageSize` = 4096
- static int `_pageCount` = 16

## 4.95.1 Macro Definition Documentation

### 4.95.1.1 LIBALLOC\_MAGIC

```
#define LIBALLOC_MAGIC 0xc001c0de
```

Durand's Ridiculously Amazing Super Duper Memory functions.

Definition at line [7](#) of file [liballoc.c](#).

### 4.95.1.2 MAXCOMPLETE

```
#define MAXCOMPLETE 5
```

Definition at line [8](#) of file [liballoc.c](#).

### 4.95.1.3 MAXEXP

```
#define MAXEXP 32
```

Definition at line [9](#) of file [liballoc.c](#).

### 4.95.1.4 MINEXP

```
#define MINEXP 8
```

Definition at line [10](#) of file [liballoc.c](#).

### 4.95.1.5 MODE

```
#define MODE MODE_BEST
```

Definition at line [15](#) of file [liballoc.c](#).

#### 4.95.1.6 MODE\_BEST

```
#define MODE_BEST 0
```

Definition at line 12 of file [liballoc.c](#).

#### 4.95.1.7 MODE\_INSTANT

```
#define MODE_INSTANT 1
```

Definition at line 13 of file [liballoc.c](#).

### 4.95.2 Function Documentation

#### 4.95.2.1 absorb\_right()

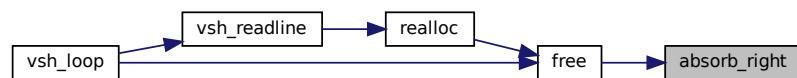
```
static struct boundary_tag* absorb_right (
    struct boundary_tag * tag ) [inline], [static]
```

Definition at line 185 of file [liballoc.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

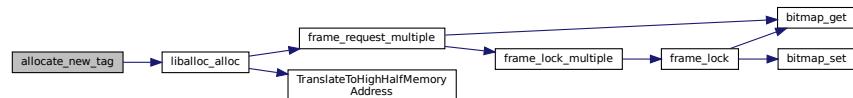


#### 4.95.2.2 allocate\_new\_tag()

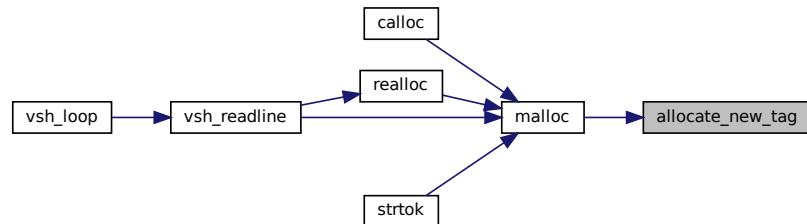
```
static struct boundary_tag* allocate_new_tag (
    unsigned int size ) [static]
```

Definition at line 229 of file [liballoc.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

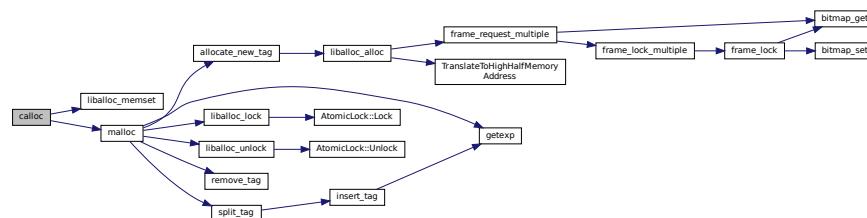


#### 4.95.2.3 calloc()

```
void* calloc (
    size_t nobj,
    size_t size )
```

Definition at line 465 of file [liballoc.c](#).

Here is the call graph for this function:

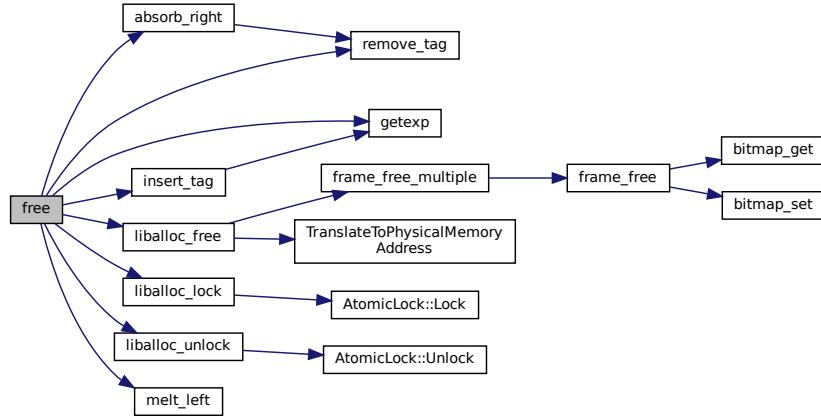


#### 4.95.2.4 free()

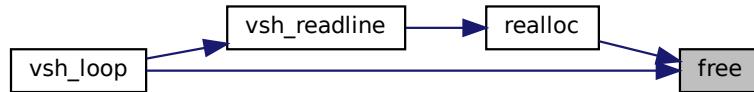
```
void free (
    void * ptr )
```

Definition at line 377 of file [liballoc.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.95.2.5 getexp()

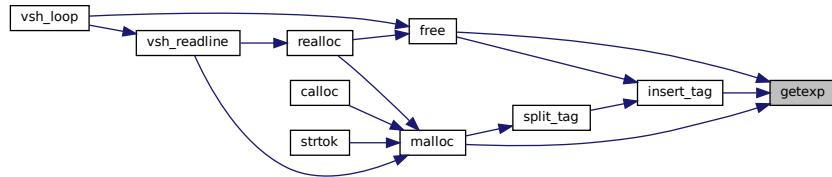
```
static int getexp (
    unsigned int size ) [inline], [static]
```

Returns the exponent required to manage 'size' amount of memory.

Returns n where  $2^n \leq \text{size} < 2^{n+1}$

Definition at line 39 of file [liballoc.c](#).

Here is the caller graph for this function:

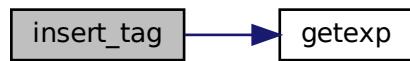


#### 4.95.2.6 `insert_tag()`

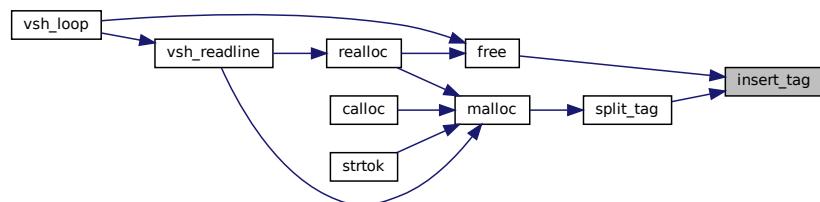
```
static void insert_tag (
    struct boundary_tag * tag,
    int index ) [inline], [static]
```

Definition at line 133 of file [liballoc.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.95.2.7 liballoc\_memcpy()

```
static void* liballoc_memcpy (
    void * s1,
    const void * s2,
    size_t n ) [static]
```

Definition at line 74 of file [liballoc.c](#).

Here is the caller graph for this function:



#### 4.95.2.8 liballoc\_memset()

```
static void* liballoc_memset (
    void * s,
    int c,
    size_t n ) [static]
```

Definition at line 65 of file [liballoc.c](#).

Here is the caller graph for this function:

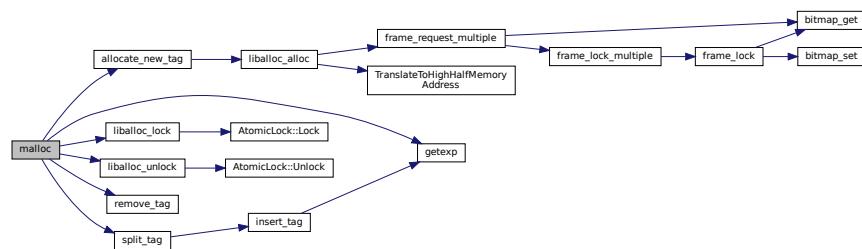


#### 4.95.2.9 malloc()

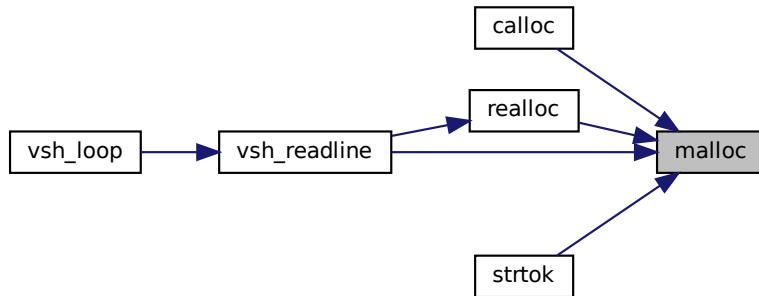
```
void* malloc (
    size_t size )
```

Definition at line 273 of file liballoc.c.

Here is the call graph for this function:



Here is the caller graph for this function:

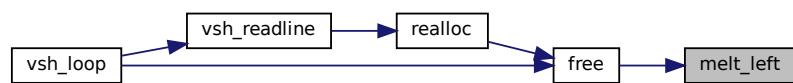


#### 4.95.2.10 melt\_left()

```
static struct boundary_tag* melt_left (
    struct boundary_tag * tag ) [inline], [static]
```

Definition at line 172 of file liballoc.c.

Here is the caller graph for this function:

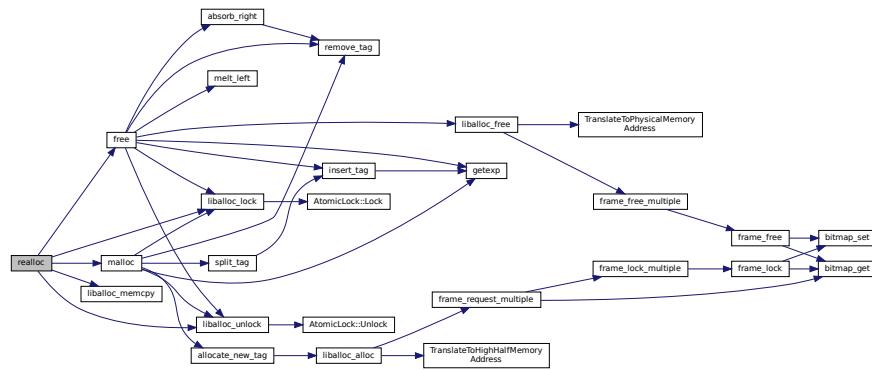


### 4.95.2.11 realloc()

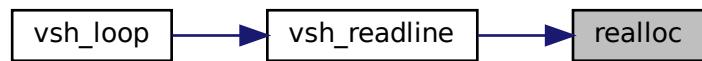
```
void* realloc (
    void * p,
    size_t size )
```

Definition at line 479 of file [liballoc.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

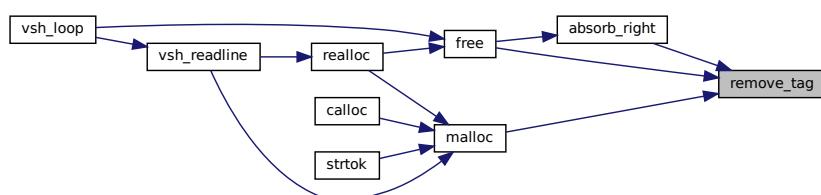


### 4.95.2.12 remove\_tag()

```
static void remove_tag (
    struct boundary_tag * tag ) [inline], [static]
```

Definition at line 157 of file [liballoc.c](#).

Here is the caller graph for this function:



#### 4.95.2.13 `split_tag()`

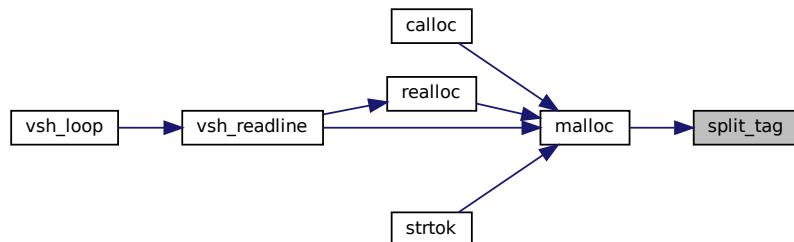
```
static struct boundary_tag* split_tag (
    struct boundary_tag * tag ) [inline], [static]
```

Definition at line 200 of file [liballoc.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 4.95.3 Variable Documentation

#### 4.95.3.1 `l_completePages`

```
int l_completePages[MAXEXP]
```

Definition at line 22 of file [liballoc.c](#).

#### 4.95.3.2 l\_freePages

```
struct boundary_tag* l_freePages[MAXEXP]
```

Definition at line 21 of file [liballoc.c](#).

#### 4.95.3.3 l\_initialized

```
int l_initialized = 0 [static]
```

Definition at line 29 of file [liballoc.c](#).

#### 4.95.3.4 l\_pageCount

```
int l_pageCount = 16 [static]
```

Definition at line 31 of file [liballoc.c](#).

#### 4.95.3.5 lPageSize

```
int lPageSize = 4096 [static]
```

Definition at line 30 of file [liballoc.c](#).

## 4.96 liballoc.c

```
00001 #include "include/liballoc.h"
00002
00005 //">#define DEBUG
00006
00007 #define LIBALLOC_MAGIC 0xc001c0de
00008 #define MAXCOMPLETE 5
00009 #define MAXEXP 32
00010 #define MINEXP 8
00011
00012 #define MODE_BEST 0
00013 #define MODE_INSTANT 1
00014
00015 #define MODE MODE_BEST
00016
00017 #ifdef DEBUG
00018 #include <stdio.h>
00019 #endif
00020
00021 struct boundary_tag *l_freePages[MAXEXP]; //< Allowing for 2^MAXEXP blocks
00022 int l_completePages[MAXEXP]; //< Allowing for 2^MAXEXP blocks
00023
00024 #ifdef DEBUG
00025 unsigned int l_allocated = 0; //< The real amount of memory allocated.
00026 unsigned int l_inuse = 0; //< The amount of memory in use (malloc'ed).
00027 #endif
00028
00029 static int l_initialized = 0; //< Flag to indicate initialization.
00030 static int lPageSize = 4096; //< Individual page size
00031 static int l_pageCount = 16; //< Minimum number of pages to allocate.
```

```

00032
00033 // **** HELPER FUNCTIONS ****
00034
00035 static inline int getexp(unsigned int size)
00036 {
00037     if (size < (1 << MINEXP))
00038     {
00039 #ifdef DEBUG
00040         printf("getexp returns -1 for %i less than MINEXP\n", size);
00041 #endif
00042         return -1; // Smaller than the quantum.
00043     }
00044     int shift = MINEXP;
00045
00046     while (shift < MAXEXP)
00047     {
00048         if ((1 << shift) > size)
00049             break;
00050         shift += 1;
00051     }
00052 #ifdef DEBUG
00053     printf("getexp returns %i (%i bytes) for %i size\n", shift - 1, (1 << (shift - 1)), size);
00054 #endif
00055     return shift - 1;
00056 }
00057
00058 static void *liballoc_memset(void *s, int c, size_t n)
00059 {
00060     int i;
00061     for (i = 0; i < n; i++)
00062         ((char *)s)[i] = c;
00063     return s;
00064 }
00065
00066 static void *liballoc_memcpy(void *s1, const void *s2, size_t n)
00067 {
00068     char *cdest;
00069     char *csrc;
00070     unsigned int *ldest = (unsigned int *)s1;
00071     unsigned int *lsrc = (unsigned int *)s2;
00072
00073     while (n >= sizeof(unsigned int))
00074     {
00075         *ldest++ = *lsrc++;
00076         n -= sizeof(unsigned int);
00077     }
00078
00079     cdest = (char *)ldest;
00080     csrc = (char *)lsrc;
00081
00082     while (n > 0)
00083     {
00084         *cdest++ = *csrc++;
00085         n -= 1;
00086     }
00087
00088     return s1;
00089 }
00090
00091 #ifdef DEBUG
00092 static void dump_array()
00093 {
00094     int i = 0;
00095     struct boundary_tag *tag = NULL;
00096
00097     printf("----- Free pages array ----- \n");
00098     printf("System memory allocated: %i\n", l_allocated);
00099     printf("Memory in used (malloc'ed): %i\n", l_inuse);
00100
00101     for (i = 0; i < MAXEXP; i++)
00102     {
00103         printf("%2i(%i): ", i, l_completePages[i]);
00104
00105         tag = l_freePages[i];
00106         while (tag != NULL)
00107         {
00108             if (tag->split_left != NULL)
00109                 printf("*");
00110             printf("%i", tag->real_size);
00111             if (tag->split_right != NULL)
00112                 printf("*");
00113
00114             tag = tag->split_right;
00115         }
00116     }
00117
00118     printf(" ");
00119 }
00120
00121
00122

```

```

00123         tag = tag->next;
00124     }
00125     printf("\n");
00126 }
00127
00128     printf("'*' denotes a split to the left/right of a tag\n");
00129     fflush(stdout);
00130 }
00131 #endif
00132
00133 static inline void insert_tag(struct boundary_tag *tag, int index)
00134 {
00135     int realIndex;
00136
00137     if (index < 0)
00138     {
00139         realIndex = getexp(tag->real_size - sizeof(struct boundary_tag));
00140         if (realIndex < MINEXP)
00141             realIndex = MINEXP;
00142     }
00143     else
00144         realIndex = index;
00145
00146     tag->index = realIndex;
00147
00148     if (l_freePages[realIndex] != NULL)
00149     {
00150         l_freePages[realIndex]->prev = tag;
00151         tag->next = l_freePages[realIndex];
00152     }
00153
00154     l_freePages[realIndex] = tag;
00155 }
00156
00157 static inline void remove_tag(struct boundary_tag *tag)
00158 {
00159     if (l_freePages[tag->index] == tag)
00160         l_freePages[tag->index] = tag->next;
00161
00162     if (tag->prev != NULL)
00163         tag->prev->next = tag->next;
00164     if (tag->next != NULL)
00165         tag->next->prev = tag->prev;
00166
00167     tag->next = NULL;
00168     tag->prev = NULL;
00169     tag->index = -1;
00170 }
00171
00172 static inline struct boundary_tag *melt_left(struct boundary_tag *tag)
00173 {
00174     struct boundary_tag *left = tag->split_left;
00175
00176     left->real_size += tag->real_size;
00177     left->split_right = tag->split_right;
00178
00179     if (tag->split_right != NULL)
00180         tag->split_right->split_left = left;
00181
00182     return left;
00183 }
00184
00185 static inline struct boundary_tag *absorb_right(struct boundary_tag *tag)
00186 {
00187     struct boundary_tag *right = tag->split_right;
00188
00189     remove_tag(right); // Remove right from free pages.
00190
00191     tag->real_size += right->real_size;
00192
00193     tag->split_right = right->split_right;
00194     if (right->split_right != NULL)
00195         right->split_right->split_left = tag;
00196
00197     return tag;
00198 }
00199
00200 static inline struct boundary_tag *split_tag(struct boundary_tag *tag)
00201 {
00202     unsigned int remainder = tag->real_size - sizeof(struct boundary_tag) - tag->size;
00203
00204     struct boundary_tag *new_tag =
00205         (struct boundary_tag *)((unsigned int)tag + sizeof(struct boundary_tag) + tag->size);
00206
00207     new_tag->magic = LIBALLOC_MAGIC;
00208     new_tag->real_size = remainder;
00209

```

```

00210     new_tag->next = NULL;
00211     new_tag->prev = NULL;
00212
00213     new_tag->split_left = tag;
00214     new_tag->split_right = tag->split_right;
00215
00216     if (new_tag->split_right != NULL)
00217         new_tag->split_right->split_left = new_tag;
00218     tag->split_right = new_tag;
00219
00220     tag->real_size -= new_tag->real_size;
00221
00222     insert_tag(new_tag, -1);
00223
00224     return new_tag;
00225 }
00226
00227 // ****
00228
00229 static struct boundary_tag *allocate_new_tag(unsigned int size)
00230 {
00231     unsigned int pages;
00232     unsigned int usage;
00233     struct boundary_tag *tag;
00234
00235     // This is how much space is required.
00236     usage = size + sizeof(struct boundary_tag);
00237
00238     // Perfect amount of space
00239     pages = usage / l_pageSize;
00240     if ((usage % l_pageSize) != 0)
00241         pages += 1;
00242
00243     // Make sure it's >= the minimum size.
00244     if (pages < l_pageCount)
00245         pages = l_pageCount;
00246
00247     tag = (struct boundary_tag *)liballoc_alloc(pages);
00248
00249     if (tag == NULL)
00250         return NULL; // uh oh, we ran out of memory.
00251
00252     tag->magic = LIBALLOC_MAGIC;
00253     tag->size = size;
00254     tag->real_size = pages * l_pageSize;
00255     tag->index = -1;
00256
00257     tag->next = NULL;
00258     tag->prev = NULL;
00259     tag->split_left = NULL;
00260     tag->split_right = NULL;
00261
00262 #ifdef DEBUG
00263     printf("Resource allocated %x of %i pages (%i bytes) for %i size.\n", tag, pages, pages *
00264         l_pageSize, size);
00265     l_allocated += pages * l_pageSize;
00266
00267     printf("Total memory usage = %i KB\n", (int)((l_allocated / (1024))));
00268 #endif
00269
00270     return tag;
00271 }
00272
00273 void *malloc(size_t size)
00274 {
00275     int index;
00276     void *ptr;
00277     struct boundary_tag *tag = NULL;
00278
00279     liballoc_lock();
00280
00281     if (l_initialized == 0)
00282     {
00283 #ifdef DEBUG
00284         printf("%s\n", "liballoc initializing.");
00285 #endif
00286         for (index = 0; index < MAXEXP; index++)
00287         {
00288             l_freePages[index] = NULL;
00289             l_completePages[index] = 0;
00290         }
00291         l_initialized = 1;
00292     }
00293
00294     index = getexp(size) + MODE;
00295     if (index < MINEXP)

```

```

00296     index = MINEXP;
00297
00298     // Find one big enough.
00299     tag = l_freePages[index]; // Start at the front of the list.
00300     while (tag != NULL)
00301     {
00302         // If there's enough space in this tag.
00303         if ((tag->real_size - sizeof(struct boundary_tag)) >= (size + sizeof(struct boundary_tag)))
00304         {
00305 #ifdef DEBUG
00306             printf("Tag search found %i >= %i\n", (tag->real_size - sizeof(struct boundary_tag)),
00307                   (size + sizeof(struct boundary_tag)));
00308         }
00309     }
00310
00311     tag = tag->next;
00312 }
00313
00314 // No page found. Make one.
00315 if (tag == NULL)
00316 {
00317     if ((tag = allocate_new_tag(size)) == NULL)
00318     {
00319         liballoc_unlock();
00320         return NULL;
00321     }
00322
00323     index = getexp(tag->real_size - sizeof(struct boundary_tag));
00324 }
00325 else
00326 {
00327     remove_tag(tag);
00328
00329     if ((tag->split_left == NULL) && (tag->split_right == NULL))
00330         l_completePages[index] -= 1;
00331 }
00332
00333 // We have a free page. Remove it from the free pages list.
00334
00335 tag->size = size;
00336
00337 // Removed... see if we can re-use the excess space.
00338
00339 #ifdef DEBUG
00340     printf("Found tag with %i bytes available (requested %i bytes, leaving %i), which has exponent: %i
00341 (%i bytes)\n", tag->real_size - sizeof(struct boundary_tag), size, tag->real_size - size -
00342 sizeof(struct boundary_tag), index, 1 << index);
00343 #endif
00344
00345     unsigned int remainder = tag->real_size - size - sizeof(struct boundary_tag) * 2; // Support a new
00346     tag + remainder
00347
00348     if (((int)(remainder) > 0) /*&& ( (tag->real_size - remainder) >= (1<<MINEXP) */)
00349     {
00350         int childIndex = getexp(remainder);
00351
00352 #ifdef DEBUG
00353         printf("Seems to be splittable: %i >= 2^%i .. %i\n", remainder, childIndex, (1 <<
00354         childIndex));
00355 #endif
00356
00357         struct boundary_tag *new_tag = split_tag(tag);
00358
00359 #ifdef DEBUG
00360         printf("Old tag has become %i bytes, new tag is now %i bytes (%i exp)\n", tag->real_size,
00361               new_tag->real_size, new_tag->index);
00362     }
00363 }
00364
00365     ptr = (void *)((unsigned int)tag + sizeof(struct boundary_tag));
00366
00367 #ifdef DEBUG
00368     l_inuse += size;
00369     printf("malloc: %x, %i, %i\n", ptr, (int)l_inuse / 1024, (int)l_allocated / 1024);
00370     dump_array();
00371 #endif
00372
00373     liballoc_unlock();
00374     return ptr;
00375 }
00376

```

```

00377 void free(void *ptr)
00378 {
00379     int index;
00380     struct boundary_tag *tag;
00381
00382     if (ptr == NULL)
00383         return;
00384
00385     liballoc_lock();
00386
00387     tag = (struct boundary_tag *) ((unsigned int)ptr - sizeof(struct boundary_tag));
00388
00389     if (tag->magic != LIBALLOC_MAGIC)
00390     {
00391         liballoc_unlock(); // release the lock
00392         return;
00393     }
00394
00395 #ifdef DEBUG
00396     l_inuse -= tag->size;
00397     printf("free: %x, %i, %i\n", ptr, (int)l_inuse / 1024, (int)l_allocated / 1024);
00398 #endif
00399
00400     // MELT LEFT...
00401     while ((tag->split_left != NULL) && (tag->split_left->index >= 0))
00402     {
00403 #ifdef DEBUG
00404         printf("Melting tag left into available memory. Left was %i, becomes %i (%i)\n",
00405             tag->split_left->real_size, tag->split_left->real_size + tag->real_size, tag->split_left->real_size);
00406 #endif
00407     tag = melt_left(tag);
00408     remove_tag(tag);
00409
00410     // MELT RIGHT...
00411     while ((tag->split_right != NULL) && (tag->split_right->index >= 0))
00412     {
00413 #ifdef DEBUG
00414         printf("Melting tag right into available memory. This was was %i, becomes %i (%i)\n",
00415             tag->real_size, tag->split_right->real_size + tag->real_size, tag->split_right->real_size);
00416 #endif
00417     tag = absorb_right(tag);
00418
00419     // Where is it going back to?
00420     index = getexp(tag->real_size - sizeof(struct boundary_tag));
00421     if (index < MINEXP)
00422         index = MINEXP;
00423
00424     // A whole, empty block?
00425     if ((tag->split_left == NULL) && (tag->split_right == NULL))
00426     {
00427
00428         if (l_completePages[index] == MAXCOMPLETE)
00429         {
00430             // Too many standing by to keep. Free this one.
00431             unsigned int pages = tag->real_size / l_pageSize;
00432
00433             if ((tag->real_size % l_pageSize) != 0)
00434                 pages += 1;
00435             if (pages < l_pageCount)
00436                 pages = l_pageCount;
00437
00438             liballoc_free(tag, pages);
00439
00440 #ifdef DEBUG
00441             l_allocated -= pages * l_pageSize;
00442             printf("Resource freeing %x of %i pages\n", tag, pages);
00443             dump_array();
00444 #endif
00445
00446         liballoc_unlock();
00447         return;
00448     }
00449
00450     l_completePages[index] += 1; // Increase the count of complete pages.
00451
00452
00453     // .....
00454
00455     insert_tag(tag, index);
00456
00457 #ifdef DEBUG
00458     printf("Returning tag with %i bytes (requested %i bytes), which has exponent: %i\n",
00459         tag->real_size, tag->size, index);
00460     dump_array();
00461 #endif

```

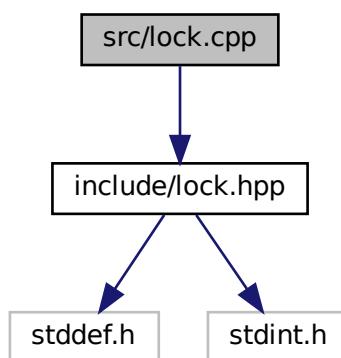
```

00461
00462     liballoc_unlock();
00463 }
00464
00465 void *calloc(size_t nobj, size_t size)
00466 {
00467     int real_size;
00468     void *p;
00469
00470     real_size = nobj * size;
00471
00472     p = malloc(real_size);
00473
00474     liballoc_memset(p, 0, real_size);
00475
00476     return p;
00477 }
00478
00479 void *realloc(void *p, size_t size)
00480 {
00481     void *ptr;
00482     struct boundary_tag *tag;
00483     int real_size;
00484
00485     if (size == 0)
00486     {
00487         free(p);
00488         return NULL;
00489     }
00490     if (p == NULL)
00491         return malloc(size);
00492
00493     if (liballoc_lock != NULL)
00494         liballoc_lock(); // lockit
00495     tag = (struct boundary_tag *)((unsigned int)p - sizeof(struct boundary_tag));
00496     real_size = tag->size;
00497     if (liballoc_unlock != NULL)
00498         liballoc_unlock();
00499
00500     if (real_size > size)
00501         real_size = size;
00502
00503     ptr = malloc(size);
00504     liballoc_memcpy(ptr, p, real_size);
00505     free(p);
00506
00507     return ptr;
00508 }

```

## 4.97 src/lock.cpp File Reference

#include "include/lock.hpp"  
Include dependency graph for lock.cpp:



## 4.98 lock.cpp

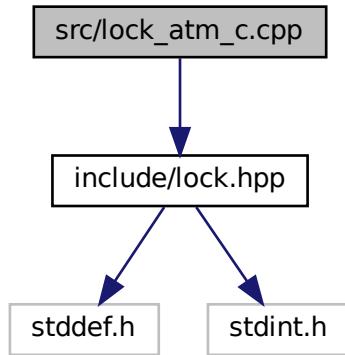
```

00001 #include "include/lock.hpp"
00002
00003 AtomicLock::AtomicLock() : locked(false) {};
00004
00005 bool AtomicLock::IsLocked() const
00006 {
00007     uint32_t result = 0;
00008
00009     __atomic_load(&locked, &result, __ATOMIC_SEQ_CST);
00010
00011     return result;
00012 }
00013
00014 void AtomicLock::Lock()
00015 {
00016     uint32_t expected = false;
00017     uint32_t desired = true;
00018
00019     while (!__atomic_compare_exchange(&locked, &expected, &desired, false, __ATOMIC_ACQUIRE,
00020           __ATOMIC_RELAXED))
00021     {
00022         expected = false;
00023
00024         asm volatile("pause");
00025     }
00026
00027 void AtomicLock::ForceLock()
00028 {
00029     locked = true;
00030 }
00031
00032 void AtomicLock::Unlock()
00033 {
00034     __atomic_store_n(&locked, false, __ATOMIC_RELEASE);
00035
00036     locked = false;
00037 }
00038
00039 ScopedLock::ScopedLock(AtomicLock &value) : lock(value)
00040 {
00041     lock.Lock();
00042 }
00043
00044 ScopedLock::~ScopedLock()
00045 {
00046     lock.Unlock();
00047 }
```

## 4.99 src/lock\_atm\_c.cpp File Reference

```
#include "include/lock.hpp"
```

Include dependency graph for lock\_atm\_c.cpp:



## Functions

- int [atomic\\_lock \(\)](#)
- int [atomic\\_unlock \(\)](#)

## Variables

- [AtomicLock c\\_lock](#)

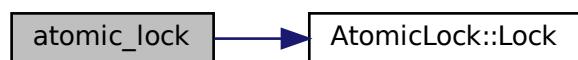
### 4.99.1 Function Documentation

#### 4.99.1.1 atomic\_lock()

```
int atomic_lock ( )
```

Definition at line 5 of file [lock\\_atm\\_c.cpp](#).

Here is the call graph for this function:

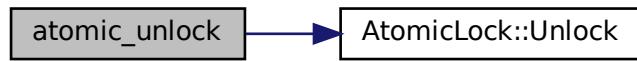


#### 4.99.1.2 atomic\_unlock()

```
int atomic_unlock ( )
```

Definition at line 13 of file [lock\\_atm\\_c.cpp](#).

Here is the call graph for this function:



#### 4.99.2 Variable Documentation

##### 4.99.2.1 c\_lock

```
AtomicLock c_lock
```

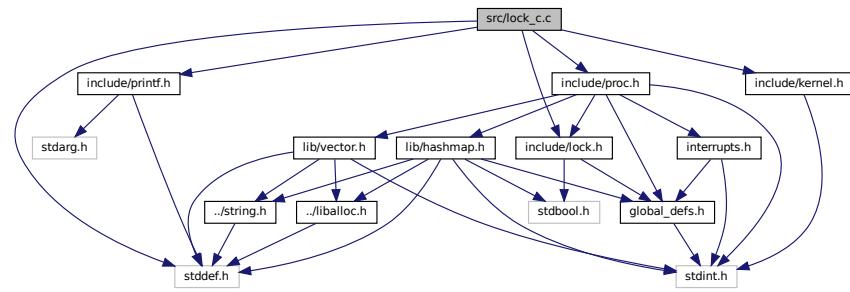
Definition at line 3 of file [lock\\_atm\\_c.cpp](#).

### 4.100 lock\_atm\_c.cpp

```
00001 #include "include/lock.hpp"
00002
00003 AtomicLock c_lock;
00004
00005 extern "C" int atomic_lock()
00006 {
00007     c_lock.Lock();
00008     return 0;
00009 }
00010
00011
00012
00013 extern "C" int atomic_unlock()
00014 {
00015     c_lock.Unlock();
00016     return 0;
00017 }
00018
00019 }
```

## 4.101 src/lock\_c.c File Reference

```
#include <stddef.h>
#include "include/lock.h"
#include "include/proc.h"
#include "include/kernel.h"
#include "include/printf.h"
Include dependency graph for lock_c.c:
```

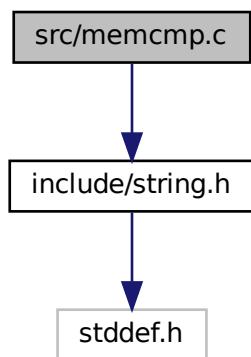


## 4.102 lock\_c.c

```
00001 #include <stddef.h>
00002 #include "include/lock.h"
00003 #include "include/proc.h"
00004 #include "include/kernel.h"
00005 #include "include/printf.h"
```

## 4.103 src/memcmp.c File Reference

```
#include "include/string.h"
Include dependency graph for memcmp.c:
```



## Functions

- int `memcmp` (const void \*aptr, const void \*bptr, size\_t size)

### 4.103.1 Function Documentation

#### 4.103.1.1 `memcmp()`

```
int memcmp (
    const void * aptr,
    const void * bptr,
    size_t size )
```

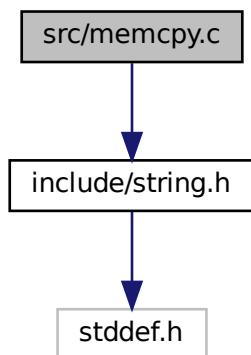
Definition at line 3 of file [memcmp.c](#).

## 4.104 memcmp.c

```
00001 #include "include/string.h"
00002
00003 int memcmp(const void *aptr, const void *bptr, size_t size)
00004 {
00005     const unsigned char *a = (const unsigned char *)aptr;
00006     const unsigned char *b = (const unsigned char *)bptr;
00007     for (size_t i = 0; i < size; i++)
00008     {
00009         if (a[i] < b[i])
00010             return -1;
00011         else if (b[i] < a[i])
00012             return 1;
00013     }
00014     return 0;
00015 }
```

## 4.105 src/memcpy.c File Reference

```
#include "include/string.h"
Include dependency graph for memcpy.c:
```



## Functions

- void \* **memcpy** (void \*restrict dstptr, const void \*restrict srcptr, size\_t size)

### 4.105.1 Function Documentation

#### 4.105.1.1 memcpy()

```
void* memcpy (
    void *restrict dstptr,
    const void *restrict srcptr,
    size_t size )
```

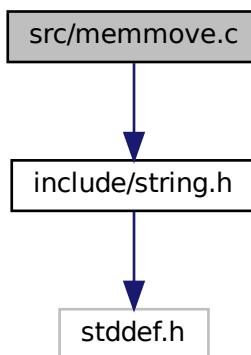
Definition at line 3 of file [memcpy.c](#).

## 4.106 memcpy.c

```
00001 #include "include/string.h"
00002
00003 void *memcpy(void *restrict dstptr, const void *restrict srcptr, size_t size)
00004 {
00005     unsigned char *dst = (unsigned char *)dstptr;
00006     const unsigned char *src = (const unsigned char *)srcptr;
00007     for (size_t i = 0; i < size; i++)
00008         dst[i] = src[i];
00009     return dstptr;
00010 }
```

## 4.107 src/memmove.c File Reference

```
#include "include/string.h"
Include dependency graph for memmove.c:
```



## Functions

- void \* **memmove** (void \*dstptr, const void \*srcptr, size\_t size)

### 4.107.1 Function Documentation

#### 4.107.1.1 memmove()

```
void* memmove (
    void * dstptr,
    const void * srcptr,
    size_t size )
```

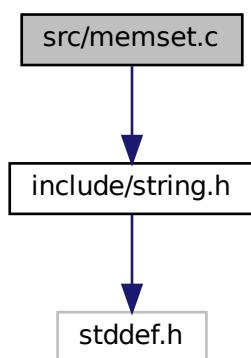
Definition at line 3 of file [memmove.c](#).

## 4.108 memmove.c

```
00001 #include "include/string.h"
00002
00003 void *memmove(void *dstptr, const void *srcptr, size_t size)
00004 {
00005     unsigned char *dst = (unsigned char *)dstptr;
00006     const unsigned char *src = (const unsigned char *)srcptr;
00007     if (dst < src)
00008     {
00009         for (size_t i = 0; i < size; i++)
00010             dst[i] = src[i];
00011     }
00012     else
00013     {
00014         for (size_t i = size; i != 0; i--)
00015             dst[i - 1] = src[i - 1];
00016     }
00017     return dstptr;
00018 }
```

## 4.109 src/memset.c File Reference

```
#include "include/string.h"
Include dependency graph for memset.c:
```



## Functions

- void \* **memset** (void \*bufptr, int value, size\_t size)

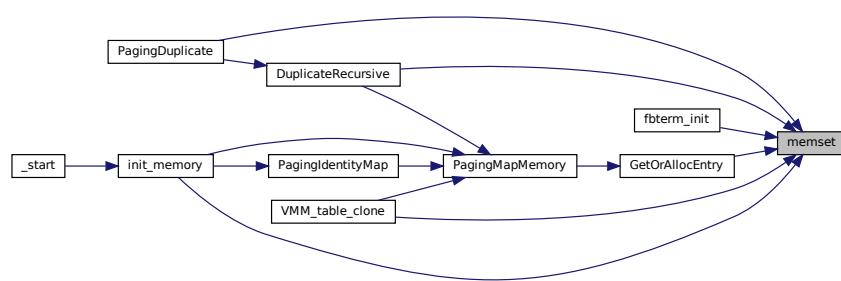
### 4.109.1 Function Documentation

#### 4.109.1.1 memset()

```
void* memset (
    void * bufptr,
    int value,
    size_t size )
```

Definition at line 3 of file [memset.c](#).

Here is the caller graph for this function:

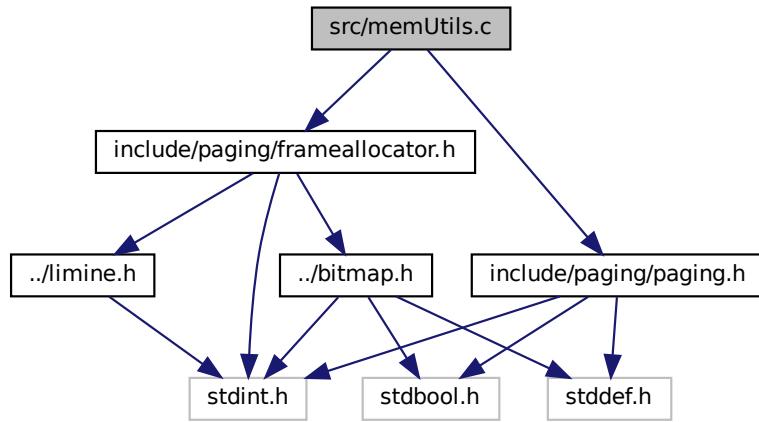


## 4.110 memset.c

```
00001 #include "include/string.h"
00002
00003 void *memset(void *bufptr, int value, size_t size)
00004 {
00005     unsigned char *buf = (unsigned char *)bufptr;
00006     for (size_t i = 0; i < size; i++)
00007         buf[i] = (unsigned char)value;
00008     return bufptr;
00009 }
```

## 4.111 src/memUtils.c File Reference

```
#include "include/paging/frameallocator.h"
#include "include/paging/paging.h"
Include dependency graph for memUtils.c:
```



### Functions

- `PAGING_EXPORT uint64_t PagingGetFreeFrame ()`

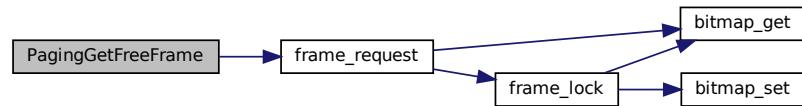
#### 4.111.1 Function Documentation

##### 4.111.1.1 PagingGetFreeFrame()

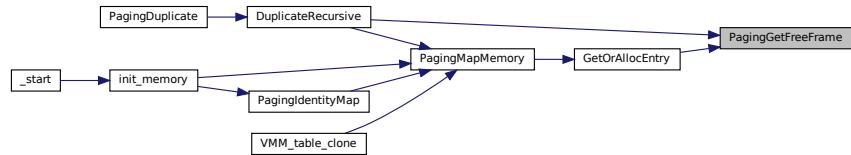
`PAGING_EXPORT uint64_t PagingGetFreeFrame ( )`

Definition at line 4 of file `memUtils.c`.

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.112 memUtils.c

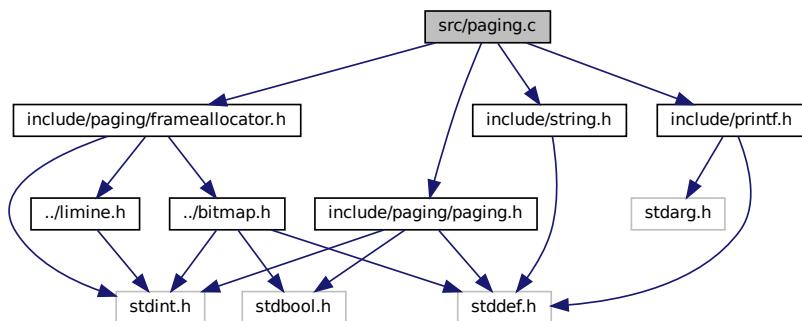
```

00001 #include "include/paging/frameallocator.h"
00002 #include "include/paging/paging.h"
00003
00004 PAGING_EXPORT uint64_t PagingGetFreeFrame()
00005 {
00006
00007     return (uint64_t) frame_request();
00008 }
  
```

## 4.113 src/paging.c File Reference

```

#include "include/paging/paging.h"
#include "include/paging/frameallocator.h"
#include "include/string.h"
#include "include/printf.h"
Include dependency graph for paging.c:
  
```



## Functions

- struct [PageTableOffset VirtualAddressToOffsets](#) (void \*virtualAddress)
- void \* [OffsetToVirtualAddress](#) (struct [PageTableOffset](#) offset)
- uint64\_t [ReadCR3](#) ()
- void [WriteCR3](#) (uint64\_t value)
- static struct [PageTable](#) \* [GetOrAllocEntry](#) (struct [PageTable](#) \*table, uint64\_t offset, uint64\_t flags)
- static struct [PageTable](#) \* [GetOrNullifyEntry](#) (struct [PageTable](#) \*table, uint64\_t offset)

- static uint64\_t [DuplicateRecursive](#) (struct PageTable \*self, uint64\_t entry, uint64\_t level)
- void [PagingIdentityMap](#) (struct PageTable \*p4, void \*virtualMemory, uint64\_t flags)
- void [PagingMapMemory](#) (struct PageTable \*p4, void \*virtualMemory, void \*physicalMemory, uint64\_t flags)
- void \* [PagingPhysicalMemory](#) (struct PageTable \*p4, void \*virtualMemory)
- void [PagingUnmapMemory](#) (struct PageTable \*p4, void \*virtualMemory)
- void [PagingDuplicate](#) (struct PageTable \*p4, struct PageTable \*newTable)

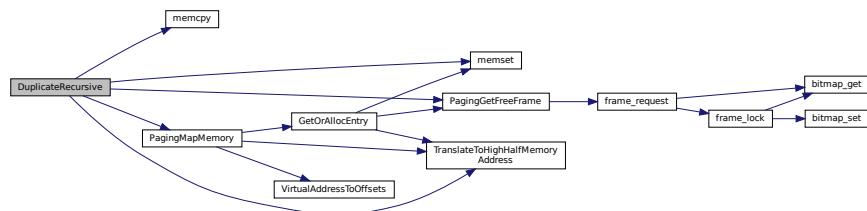
## 4.113.1 Function Documentation

### 4.113.1.1 [DuplicateRecursive\(\)](#)

```
static uint64_t DuplicateRecursive (
    struct PageTable * self,
    uint64_t entry,
    uint64_t level ) [inline], [static]
```

Definition at line 85 of file [paging.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

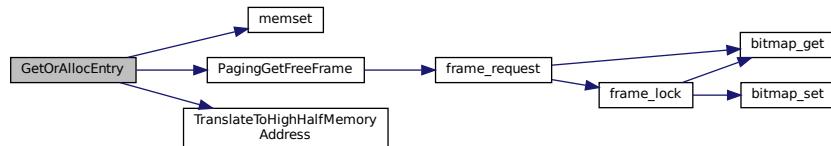


### 4.113.1.2 GetOrAllocEntry()

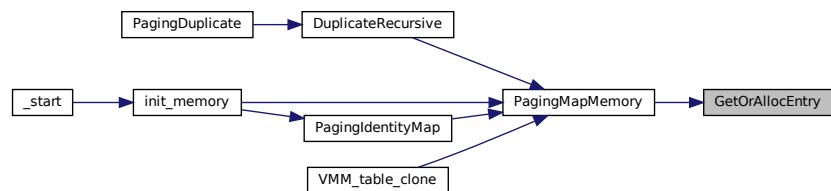
```
static struct PageTable* GetOrAllocEntry (
    struct PageTable * table,
    uint64_t offset,
    uint64_t flags ) [inline], [static]
```

Definition at line 52 of file [paging.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

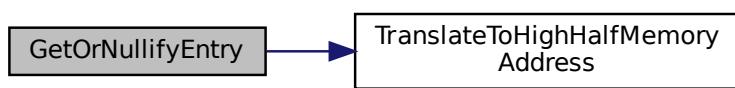


### 4.113.1.3 GetOrNullifyEntry()

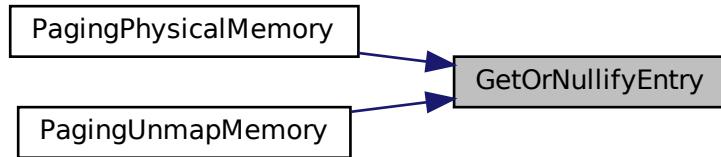
```
static struct PageTable* GetOrNullifyEntry (
    struct PageTable * table,
    uint64_t offset ) [inline], [static]
```

Definition at line 73 of file [paging.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.113.1.4 OffsetToVirtualAddress()

```
void* OffsetToVirtualAddress (
    struct PageTableOffset offset )
```

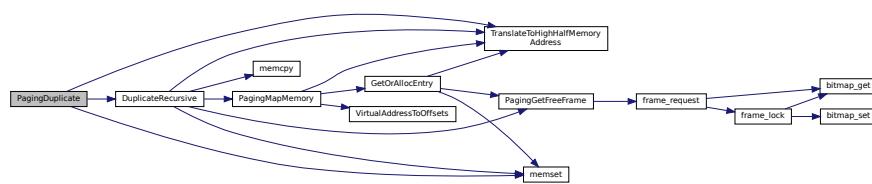
Definition at line 21 of file [paging.c](#).

#### 4.113.1.5 PagingDuplicate()

```
void PagingDuplicate (
    struct PageTable * p4,
    struct PageTable * newTable )
```

Definition at line 194 of file [paging.c](#).

Here is the call graph for this function:

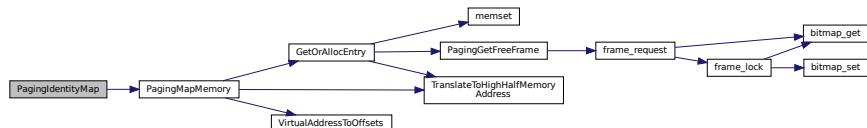


#### 4.113.1.6 PagingIdentityMap()

```
void PagingIdentityMap (
    struct PageTable * p4,
    void * virtualMemory,
    uint64_t flags )
```

Definition at line 115 of file [paging.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

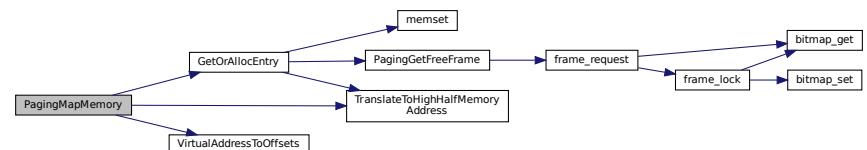


#### 4.113.1.7 PagingMapMemory()

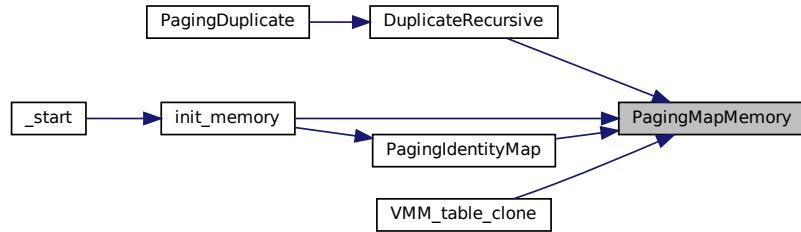
```
void PagingMapMemory (
    struct PageTable * p4,
    void * virtualMemory,
    void * physicalMemory,
    uint64_t flags )
```

Definition at line 120 of file [paging.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

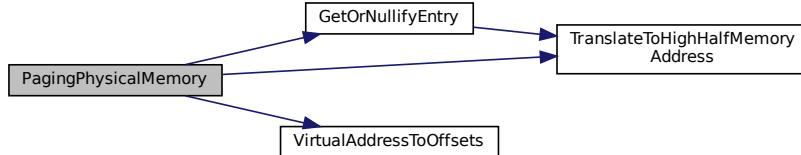


#### 4.113.1.8 PagingPhysicalMemory()

```
void* PagingPhysicalMemory (
    struct PageTable * p4,
    void * virtualMemory )
```

Definition at line 135 of file [paging.c](#).

Here is the call graph for this function:

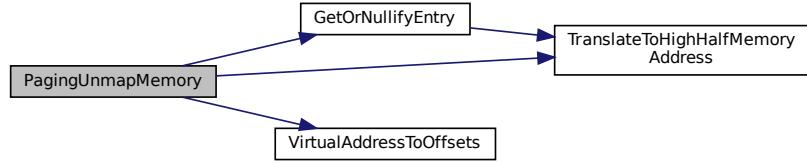


#### 4.113.1.9 PagingUnmapMemory()

```
void PagingUnmapMemory (
    struct PageTable * p4,
    void * virtualMemory )
```

Definition at line 165 of file [paging.c](#).

Here is the call graph for this function:



#### 4.113.1.10 ReadCR3()

```
uint64_t ReadCR3 ( )
```

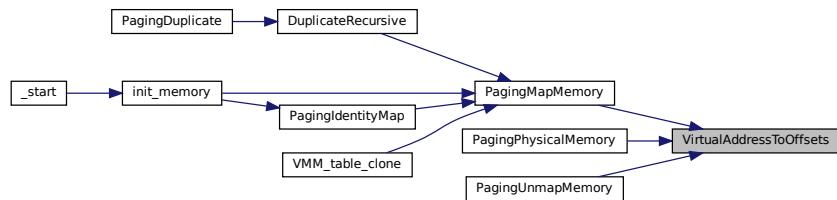
Definition at line 33 of file [paging.c](#).

#### 4.113.1.11 VirtualAddressToOffsets()

```
struct PageTableOffset VirtualAddressToOffsets (
    void * virtualAddress )
```

Definition at line 1 of file [paging.c](#).

Here is the caller graph for this function:



#### 4.113.1.12 WriteCR3()

```
void WriteCR3 (
    uint64_t value )
```

Definition at line 45 of file [paging.c](#).

## 4.114 paging.c

```

00001 #include "include/paging/paging.h"
00002 #include "include/paging/frameallocator.h"
00003 #include "include/string.h"
00004 #include "include/printf.h"
00005
00006 struct PageTableOffset VirtualAddressToOffsets(void *virtualAddress)
00007 {
00008     uint64_t address = (uint64_t)virtualAddress;
00009
00010     struct PageTableOffset offset = {
00011
00012         .p4Offset = (address & ((uint64_t)0x1FF << 39)) >> 39,
00013         .pdpOffset = (address & ((uint64_t)0x1FF << 30)) >> 30,
00014         .pdOffset = (address & ((uint64_t)0x1FF << 21)) >> 21,
00015         .ptOffset = (address & ((uint64_t)0x1FF << 12)) >> 12,
00016     };
00017
00018     return offset;
00019 }
00020
00021 void *OffsetToVirtualAddress(struct PageTableOffset offset)
00022 {
00023     uint64_t address = 0;
00024
00025     address |= offset.p4Offset << 39;
00026     address |= offset.pdpOffset << 30;
00027     address |= offset.pdOffset << 21;
00028     address |= offset.ptOffset << 12;
00029
00030     return (void *)address;
00031 }
00032
00033 uint64_t ReadCR3()
00034 {
00035     uint64_t outValue = 0;
00036
00037     asm("mov %%cr3, %0"
00038         : "=r"(outValue)
00039         :);
00040
00041     return outValue;
00042     // no input
00043 }
00044
00045 void WriteCR3(uint64_t value)
00046 {
00047     asm("mov %0, %%cr3"
00048         : // no output
00049         : "r"(value));
00050 }
00051
00052 static inline struct PageTable *GetOrAllocEntry(struct PageTable *table, uint64_t offset, uint64_t flags)
00053 {
00054     uint64_t address = table->entries[offset];
00055
00056     if (!(address & PAGING_FLAG_PRESENT))
00057     {
00058         address = table->entries[offset] = PagingGetFreeFrame();
00059
00060         if (!address)
00061         {
00062             return NULL;
00063         }
00064
00065         table->entries[offset] |= flags | PAGING_FLAG_PRESENT;
00066
00067         memset((void *)TranslateToHighHalfMemoryAddress(address), 0, 0x1000);
00068     }
00069
00070     return (struct PageTable *)TranslateToHighHalfMemoryAddress(address & PAGE_ADDRESS_MASK);
00071 }
00072
00073 static inline struct PageTable *GetOrNullifyEntry(struct PageTable *table, uint64_t offset)
00074 {
00075     uint64_t address = table->entries[offset];
00076
00077     if (!(address & PAGING_FLAG_PRESENT))
00078     {
00079         return NULL;
00080     }
00081
00082     return (struct PageTable *)TranslateToHighHalfMemoryAddress(address & PAGE_ADDRESS_MASK);
00083 }
00084

```

```

00085 static inline uint64_t DuplicateRecursive(struct PageTable *self, uint64_t entry, uint64_t level)
00086 {
00087     const uint64_t flags = PAGING_FLAG_PRESENT | PAGING_FLAG_USER_ACCESSIBLE | PAGING_FLAG_WRITABLE;
00088
00089     uint64_t *virt = (uint64_t *)TranslateToHighHalfMemoryAddress((entry & ~PAGE_FLAG_MASK));
00090     uint64_t newPage = PagingGetFreeFrame();
00091     uint64_t *newVirtual = (uint64_t *)TranslateToHighHalfMemoryAddress(newPage);
00092
00093     PagingMapMemory(self, newVirtual, (void *)newPage, flags);
00094
00095     memset(newVirtual, 0, 0x1000);
00096
00097     if (level == 0)
00098     {
00099         memcpy(newVirtual, (void *)virt, 0x1000);
00100     }
00101     else
00102     {
00103         for (uint64_t i = 0; i < 512; i++)
00104         {
00105             if (virt[i] & PAGING_FLAG_PRESENT)
00106             {
00107                 newVirtual[i] = DuplicateRecursive(self, virt[i], level - 1);
00108             }
00109         }
00110     }
00111
00112     return newPage | (entry & PAGE_FLAG_MASK);
00113 }
00114
00115 void PagingIdentityMap(struct PageTable *p4, void *virtualMemory, uint64_t flags)
00116 {
00117     PagingMapMemory(p4, virtualMemory, virtualMemory, flags);
00118 }
00119
00120 void PagingMapMemory(struct PageTable *p4, void *virtualMemory, void *physicalMemory, uint64_t flags)
00121 {
00122     uint64_t higherPermissions = PAGING_FLAG_WRITABLE | PAGING_FLAG_USER_ACCESSIBLE;
00123
00124     struct PageTableOffset offset = VirtualAddressToOffsets(virtualMemory);
00125
00126     struct PageTable *p4Virtual = (struct PageTable *)TranslateToHighHalfMemoryAddress((uint64_t)p4);
00127
00128     struct PageTable *pdp = GetOrAllocEntry(p4Virtual, offset.p4Offset, higherPermissions);
00129     struct PageTable *pd = GetOrAllocEntry(pdp, offset.pdOffset, higherPermissions);
00130     struct PageTable *pt = GetOrAllocEntry(pd, offset.ptOffset, higherPermissions);
00131
00132     pt->entries[offset.ptOffset] = (uint64_t)physicalMemory | flags | PAGING_FLAG_PRESENT;
00133 }
00134
00135 void *PagingPhysicalMemory(struct PageTable *p4, void *virtualMemory)
00136 {
00137     struct PageTableOffset offset = VirtualAddressToOffsets(virtualMemory);
00138
00139     struct PageTable *p4Virtual = (struct PageTable *)TranslateToHighHalfMemoryAddress((uint64_t)p4);
00140
00141     struct PageTable *pdp = GetOrNullifyEntry(p4Virtual, offset.p4Offset);
00142
00143     if (pdp == NULL)
00144     {
00145         return NULL;
00146     }
00147
00148     struct PageTable *pd = GetOrNullifyEntry(pdp, offset.pdOffset);
00149
00150     if (pd == NULL)
00151     {
00152         return NULL;
00153     }
00154
00155     struct PageTable *pt = GetOrNullifyEntry(pd, offset.ptOffset);
00156
00157     if (pt == NULL)
00158     {
00159         return NULL;
00160     }
00161
00162     return (void *) (pt->entries[offset.ptOffset] & ~(PAGE_FLAG_MASK));
00163 }
00164
00165 void PagingUnmapMemory(struct PageTable *p4, void *virtualMemory)
00166 {
00167     struct PageTableOffset offset = VirtualAddressToOffsets(virtualMemory);
00168
00169     struct PageTable *p4Virtual = (struct PageTable *)TranslateToHighHalfMemoryAddress((uint64_t)p4);
00170     struct PageTable *pdp = GetOrNullifyEntry(p4Virtual, offset.p4Offset);
00171

```

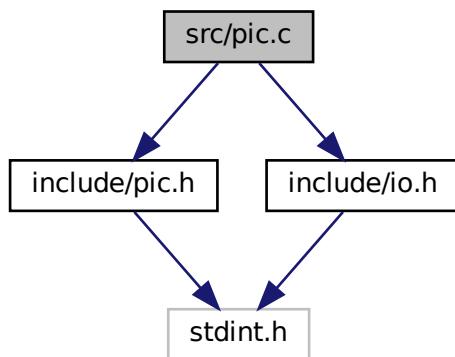
```

00172     if (pdp == NULL)
00173     {
00174         return;
00175     }
00176
00177     struct PageTable *pd = GetOrNullifyEntry(pdp, offset.pdpOffset);
00178
00179     if (pd == NULL)
00180     {
00181         return;
00182     }
00183
00184     struct PageTable *pt = GetOrNullifyEntry(pd, offset.pdOffset);
00185
00186     if (pt == NULL)
00187     {
00188         return;
00189     }
00190
00191     pt->entries[offset.ptOffset] = 0;
00192 }
00193
00194 void PagingDuplicate(struct PageTable *p4, struct PageTable *newTable)
00195 {
00196     struct PageTable *p4Virtual = (struct PageTable *)TranslateToHighHalfMemoryAddress((uint64_t)p4);
00197
00198     memset(newTable, 0, 0x1000);
00199
00200     for (uint64_t i = 0; i < 256; i++)
00201     {
00202         uint64_t entry = p4Virtual->entries[i];
00203
00204         // printf_("0x%llx\n", entry);
00205
00206         if (entry & PAGING_FLAG_PRESENT)
00207         {
00208             newTable->entries[i] = DuplicateRecursive(p4, entry, 3);
00209         }
00210     }
00211
00212     for (uint64_t i = 256; i < 512; i++)
00213     {
00214         newTable->entries[i] = p4Virtual->entries[i];
00215     }
00216 }

```

## 4.115 src/pic.c File Reference

```
#include "include/pic.h"
#include "include/io.h"
Include dependency graph for pic.c:
```



## Functions

- void [pic\\_mask\\_irq](#) (uint8\_t irq)
- void [pic\\_unmask\\_irq](#) (uint8\_t irq)
- void [pic\\_remap\\_offsets](#) (uint8\_t offset)
- void [pic\\_send\\_eoi](#) (uint8\_t irq)
- void [pic\\_enable](#) ()

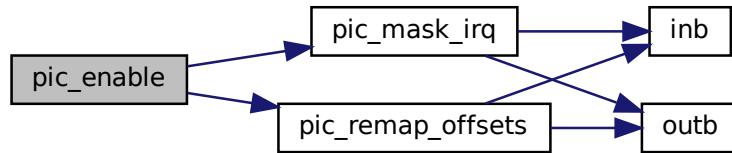
### 4.115.1 Function Documentation

#### 4.115.1.1 [pic\\_enable\(\)](#)

```
void pic_enable (
    void )
```

Definition at line [74](#) of file [pic.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

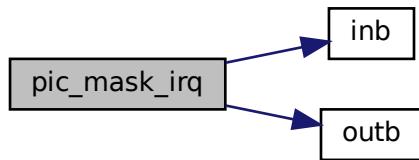


#### 4.115.1.2 pic\_mask\_irq()

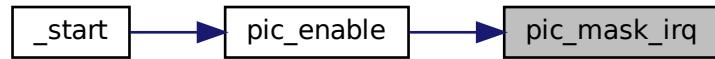
```
void pic_mask_irq (
    uint8_t irq )
```

Definition at line 4 of file [pic.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

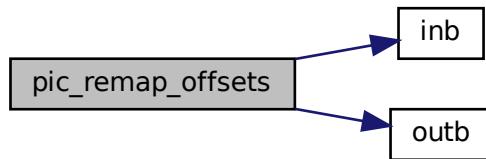


#### 4.115.1.3 pic\_remap\_offsets()

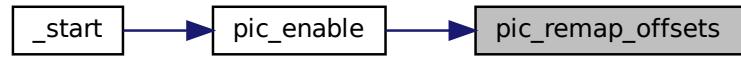
```
void pic_remap_offsets (
    uint8_t offset )
```

Definition at line 44 of file [pic.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

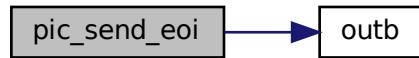


#### 4.115.1.4 pic\_send\_eoi()

```
void pic_send_eoi (
    uint8_t irq )
```

Definition at line 67 of file [pic.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

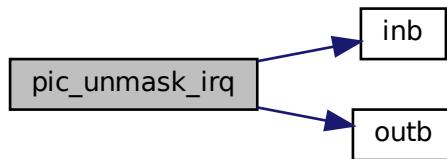


#### 4.115.1.5 pic\_unmask\_irq()

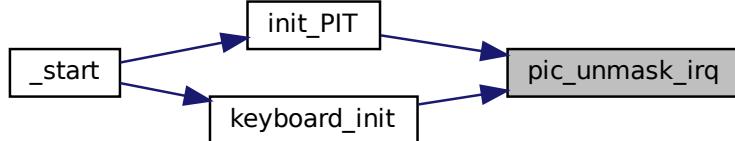
```
void pic_unmask_irq (
    uint8_t irq )
```

Definition at line 24 of file [pic.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.116 pic.c

```
00001 #include "include/pic.h"
00002 #include "include/io.h"
00003
00004 void pic_mask_irq(uint8_t irq)
00005 {
00006     uint16_t port;
00007     uint8_t masks;
00008
00009     if (irq < 8)
00010     {
00011         port = PIC_MASTER_DATA;
00012     }
00013     else
00014     {
00015         port = PIC_SLAVE_DATA;
00016         irq -= 8;
00017     }
00018
00019     masks = inb(port);
00020     masks |= (1 << irq);
00021     outb(port, masks);
00022 }
00023
00024 void pic_unmask_irq(uint8_t irq)
```

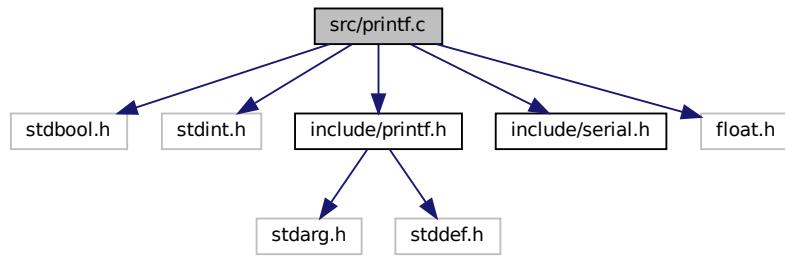
```

00025 {
00026     uint16_t port;
00027     uint8_t masks;
00028
00029     if (irq < 8)
00030     {
00031         port = PIC_MASTER_DATA;
00032     }
00033     else
00034     {
00035         port = PIC_SLAVE_DATA;
00036         irq -= 8;
00037     }
00038
00039     masks = inb(port);
00040     masks &= ~(1 << irq);
00041     outb(port, masks);
00042 }
00043
00044 void pic_remap_offsets(uint8_t offset)
00045 {
00046     uint8_t master_mask, slave_mask;
00047
00048     master_mask = inb(PIC_MASTER_DATA);
00049     slave_mask = inb(PIC_SLAVE_DATA);
00050
00051     outb(PIC_MASTER_COMMAND, PIC_ICW1_INIT | PIC_ICW1_ICW4);
00052     outb(PIC_SLAVE_COMMAND, PIC_ICW1_INIT | PIC_ICW1_ICW4);
00053
00054     outb(PIC_MASTER_DATA, offset);
00055     outb(PIC_SLAVE_DATA, offset + 0x08);
00056
00057     outb(PIC_MASTER_DATA, 0x04);
00058     outb(PIC_SLAVE_DATA, 0x02);
00059
00060     outb(PIC_MASTER_DATA, PIC_ICW4_8086);
00061     outb(PIC_SLAVE_DATA, PIC_ICW4_8086);
00062
00063     outb(PIC_MASTER_DATA, master_mask);
00064     outb(PIC_SLAVE_DATA, slave_mask);
00065 }
00066
00067 void pic_send_eoi(uint8_t irq)
00068 {
00069     if (irq >= 8)
00070         outb(PIC_SLAVE_COMMAND, PIC_EOI);
00071     outb(PIC_MASTER_COMMAND, PIC_EOI);
00072 }
00073
00074 void pic_enable()
00075 {
00076     pic_remap_offsets(0x20);
00077     for (uint8_t irq = 0; irq < 16; irq++)
00078         pic_mask_irq(irq);
00079 }
```

## 4.117 src/printf.c File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include "include/printf.h"
#include "include/serial.h"
#include <float.h>
```

Include dependency graph for printf.c:



## Data Structures

- struct [out\\_fct\\_wrap\\_type](#)

## Macros

- #define PRINTF\_NTOA\_BUFFER\_SIZE 32U
- #define PRINTF\_FTOA\_BUFFER\_SIZE 32U
- #define PRINTF\_SUPPORT\_FLOAT
- #define PRINTF\_SUPPORT\_EXPONENTIAL
- #define PRINTF\_DEFAULT\_FLOAT\_PRECISION 6U
- #define PRINTF\_MAX\_FLOAT 1e9
- #define PRINTF\_SUPPORT\_LONG\_LONG
- #define PRINTF\_SUPPORT\_PTRDIFF\_T
- #define FLAGS\_ZEROPAD (1U << 0U)
- #define FLAGS\_LEFT (1U << 1U)
- #define FLAGS\_PLUS (1U << 2U)
- #define FLAGS\_SPACE (1U << 3U)
- #define FLAGS\_HASH (1U << 4U)
- #define FLAGS\_UPPERCASE (1U << 5U)
- #define FLAGS\_CHAR (1U << 6U)
- #define FLAGS\_SHORT (1U << 7U)
- #define FLAGS\_LONG (1U << 8U)
- #define FLAGS\_LONG\_LONG (1U << 9U)
- #define FLAGS\_PRECISION (1U << 10U)
- #define FLAGS\_ADAPT\_EXP (1U << 11U)

## Typedefs

- typedef void(\* [out\\_fct\\_type](#)) (char character, void \*buffer, size\_t idx, size\_t maxlen)

## Functions

- static void [\\_out\\_buffer](#) (char character, void \*buffer, size\_t idx, size\_t maxlen)
- static void [\\_out\\_null](#) (char character, void \*buffer, size\_t idx, size\_t maxlen)
- static void [\\_out\\_char](#) (char character, void \*buffer, size\_t idx, size\_t maxlen)
- static void [\\_out\\_fct](#) (char character, void \*buffer, size\_t idx, size\_t maxlen)
- static unsigned int [\\_strnlen\\_s](#) (const char \*str, size\_t maxsize)
- static bool [\\_is\\_digit](#) (char ch)
- static unsigned int [\\_atoi](#) (const char \*\*str)
- static size\_t [\\_out\\_rev](#) ([out\\_fct\\_type](#) out, char \*buffer, size\_t idx, size\_t maxlen, const char \*buf, size\_t len, unsigned int width, unsigned int flags)
- static size\_t [\\_ntoa\\_format](#) ([out\\_fct\\_type](#) out, char \*buffer, size\_t idx, size\_t maxlen, char \*buf, size\_t len, bool negative, unsigned int base, unsigned int prec, unsigned int width, unsigned int flags)
- static size\_t [\\_ntoa\\_long](#) ([out\\_fct\\_type](#) out, char \*buffer, size\_t idx, size\_t maxlen, unsigned long value, bool negative, unsigned long base, unsigned int prec, unsigned int width, unsigned int flags)
- static size\_t [\\_ntoa\\_long\\_long](#) ([out\\_fct\\_type](#) out, char \*buffer, size\_t idx, size\_t maxlen, unsigned long long value, bool negative, unsigned long long base, unsigned int prec, unsigned int width, unsigned int flags)
- static size\_t [\\_etoa](#) ([out\\_fct\\_type](#) out, char \*buffer, size\_t idx, size\_t maxlen, double value, unsigned int prec, unsigned int width, unsigned int flags)
- static size\_t [\\_ftoa](#) ([out\\_fct\\_type](#) out, char \*buffer, size\_t idx, size\_t maxlen, double value, unsigned int prec, unsigned int width, unsigned int flags)
- static int [\\_vsnprintf](#) ([out\\_fct\\_type](#) out, char \*buffer, const size\_t maxlen, const char \*format, va\_list va)
- int [printf\\_](#) (const char \*format,...)
- int [sprintf\\_](#) (char \*buffer, const char \*format,...)
- int [snprintf\\_](#) (char \*buffer, size\_t count, const char \*format,...)
- int [vprintf\\_](#) (const char \*format, va\_list va)
- int [vsnprintf\\_](#) (char \*buffer, size\_t count, const char \*format, va\_list va)
- int [fscanf](#) (void(\*out)(char character, void \*arg), void \*arg, const char \*format,...)

### 4.117.1 Macro Definition Documentation

#### 4.117.1.1 FLAGS\_ADAPTER\_EXP

```
#define FLAGS_ADAPTER_EXP (1U << 11U)
```

Definition at line 111 of file [printf.c](#).

#### 4.117.1.2 FLAGS\_CHAR

```
#define FLAGS_CHAR (1U << 6U)
```

Definition at line 106 of file [printf.c](#).

#### 4.117.1.3 **FLAGS\_HASH**

```
#define FLAGS_HASH (1U << 4U)
```

Definition at line 104 of file [printf.c](#).

#### 4.117.1.4 **FLAGS\_LEFT**

```
#define FLAGS_LEFT (1U << 1U)
```

Definition at line 101 of file [printf.c](#).

#### 4.117.1.5 **FLAGS\_LONG**

```
#define FLAGS_LONG (1U << 8U)
```

Definition at line 108 of file [printf.c](#).

#### 4.117.1.6 **FLAGS\_LONG\_LONG**

```
#define FLAGS_LONG_LONG (1U << 9U)
```

Definition at line 109 of file [printf.c](#).

#### 4.117.1.7 **FLAGS\_PLUS**

```
#define FLAGS_PLUS (1U << 2U)
```

Definition at line 102 of file [printf.c](#).

#### 4.117.1.8 **FLAGS\_PRECISION**

```
#define FLAGS_PRECISION (1U << 10U)
```

Definition at line 110 of file [printf.c](#).

#### 4.117.1.9 FLAGS\_SHORT

```
#define FLAGS_SHORT (1U << 7U)
```

Definition at line 107 of file [printf.c](#).

#### 4.117.1.10 FLAGS\_SPACE

```
#define FLAGS_SPACE (1U << 3U)
```

Definition at line 103 of file [printf.c](#).

#### 4.117.1.11 FLAGS\_UPPERCASE

```
#define FLAGS_UPPERCASE (1U << 5U)
```

Definition at line 105 of file [printf.c](#).

#### 4.117.1.12 FLAGS\_ZEROPAD

```
#define FLAGS_ZEROPAD (1U << 0U)
```

Definition at line 100 of file [printf.c](#).

#### 4.117.1.13 PRINTF\_DEFAULT\_FLOAT\_PRECISION

```
#define PRINTF_DEFAULT_FLOAT_PRECISION 6U
```

Definition at line 75 of file [printf.c](#).

#### 4.117.1.14 PRINTF\_FTOA\_BUFFER\_SIZE

```
#define PRINTF_FTOA_BUFFER_SIZE 32U
```

Definition at line 57 of file [printf.c](#).

#### 4.117.1.15 PRINTF\_MAX\_FLOAT

```
#define PRINTF_MAX_FLOAT 1e9
```

Definition at line [81](#) of file [printf.c](#).

#### 4.117.1.16 PRINTF\_NTOA\_BUFFER\_SIZE

```
#define PRINTF_NTOA_BUFFER_SIZE 32U
```

Definition at line [50](#) of file [printf.c](#).

#### 4.117.1.17 PRINTF\_SUPPORT\_EXPONENTIAL

```
#define PRINTF_SUPPORT_EXPONENTIAL
```

Definition at line [69](#) of file [printf.c](#).

#### 4.117.1.18 PRINTF\_SUPPORT\_FLOAT

```
#define PRINTF_SUPPORT_FLOAT
```

Definition at line [63](#) of file [printf.c](#).

#### 4.117.1.19 PRINTF\_SUPPORT\_LONG\_LONG

```
#define PRINTF_SUPPORT_LONG_LONG
```

Definition at line [87](#) of file [printf.c](#).

#### 4.117.1.20 PRINTF\_SUPPORT\_PTRDIFF\_T

```
#define PRINTF_SUPPORT_PTRDIFF_T
```

Definition at line [94](#) of file [printf.c](#).

## 4.117.2 Typedef Documentation

### 4.117.2.1 out\_fct\_type

```
typedef void(* out_fct_type) (char character, void *buffer, size_t idx, size_t maxlen)
```

Definition at line 119 of file [printf.c](#).

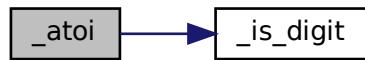
## 4.117.3 Function Documentation

### 4.117.3.1 \_atoi()

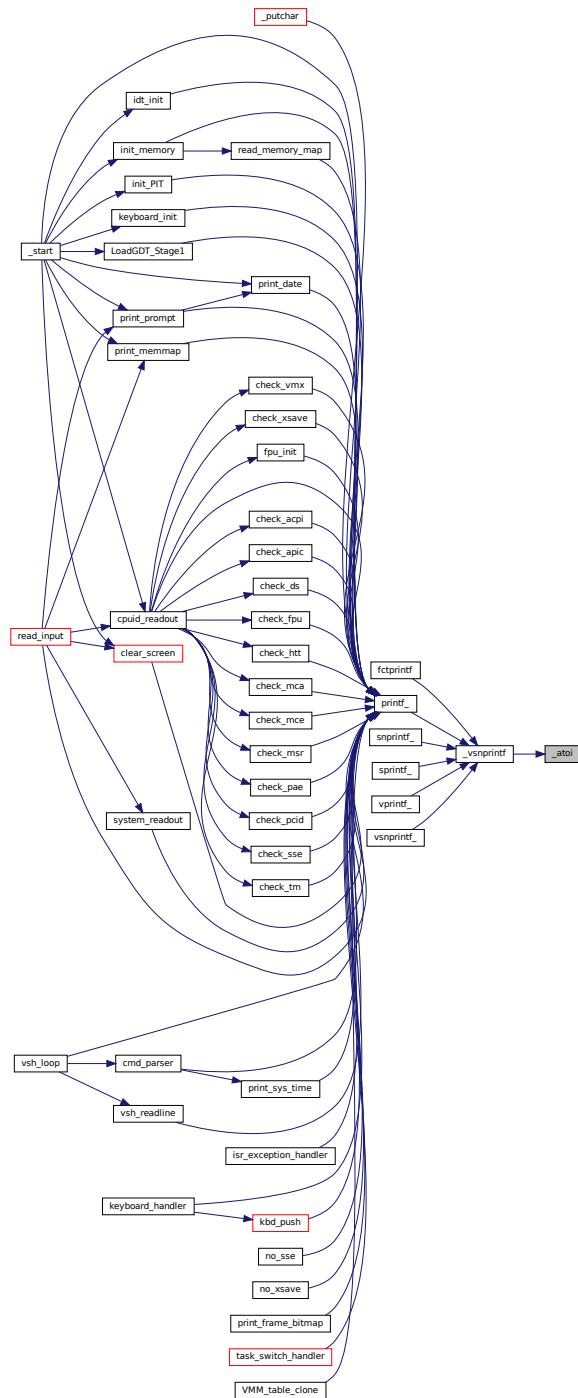
```
static unsigned int _atoi (
    const char ** str ) [static]
```

Definition at line 188 of file [printf.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



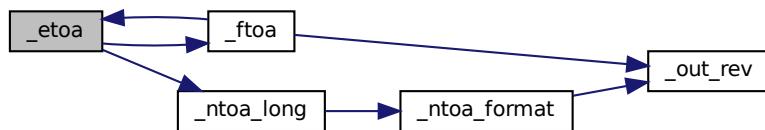
#### 4.117.3.2 `_etoa()`

```
static size_t _etoa (
    out_fct_type out,
```

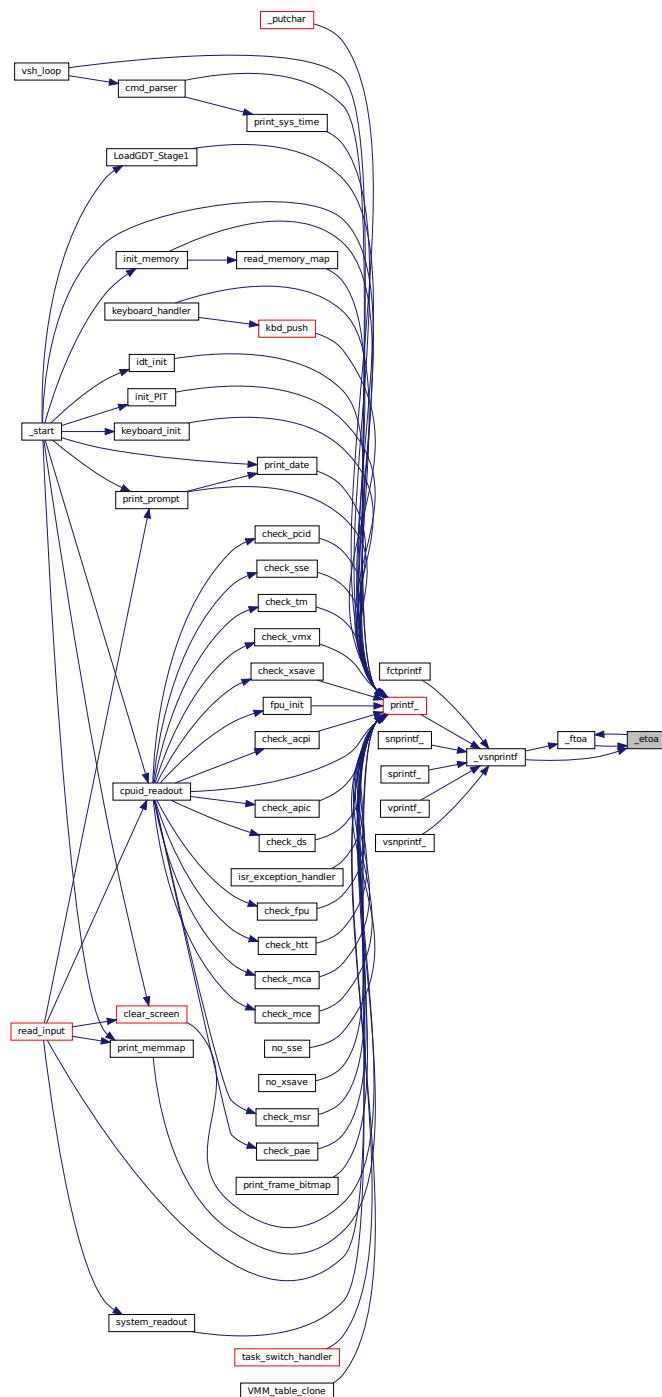
```
char * buffer,  
size_t idx,  
size_t maxlen,  
double value,  
unsigned int prec,  
unsigned int width,  
unsigned int flags ) [static]
```

Definition at line 511 of file [printf.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



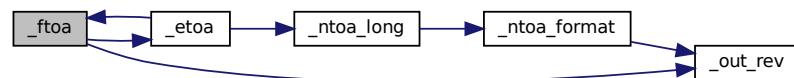
#### 4.117.3.3 `_ftoa()`

```
static size_t _ftoa (
    out_fct_type out,
```

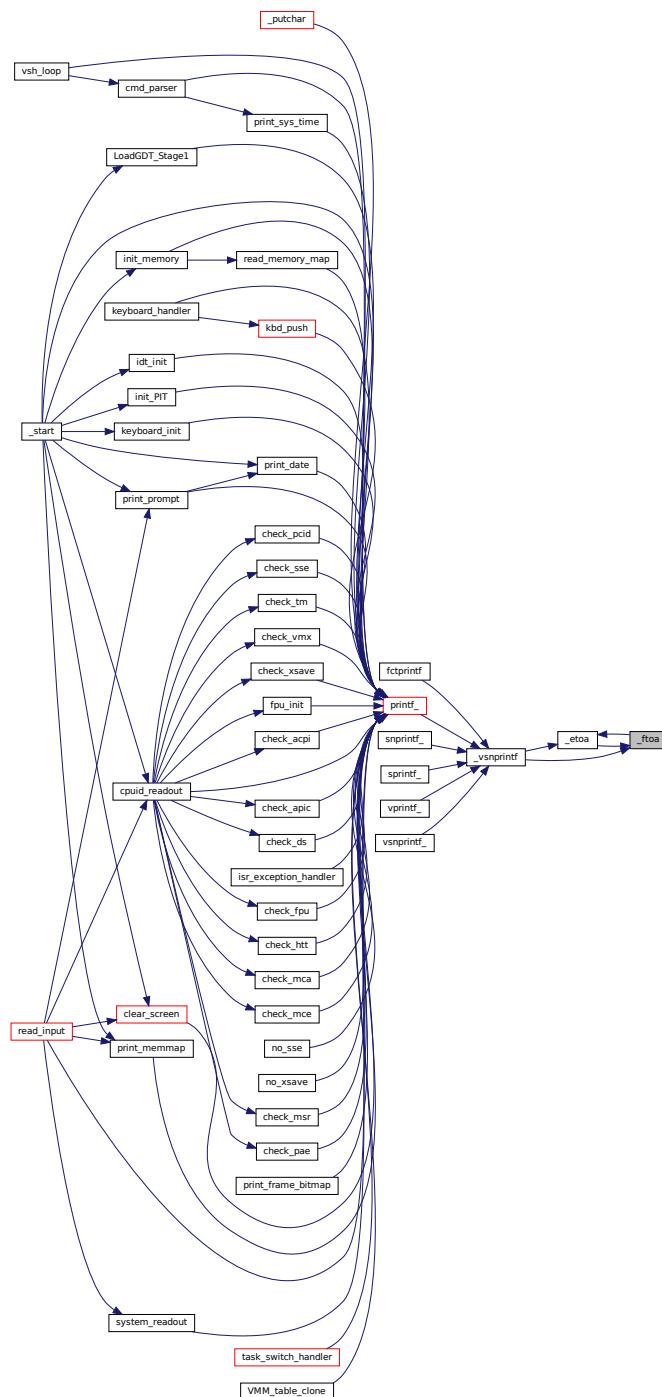
```
char * buffer,  
size_t idx,  
size_t maxlen,  
double value,  
unsigned int prec,  
unsigned int width,  
unsigned int flags ) [static]
```

Definition at line 360 of file [printf.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

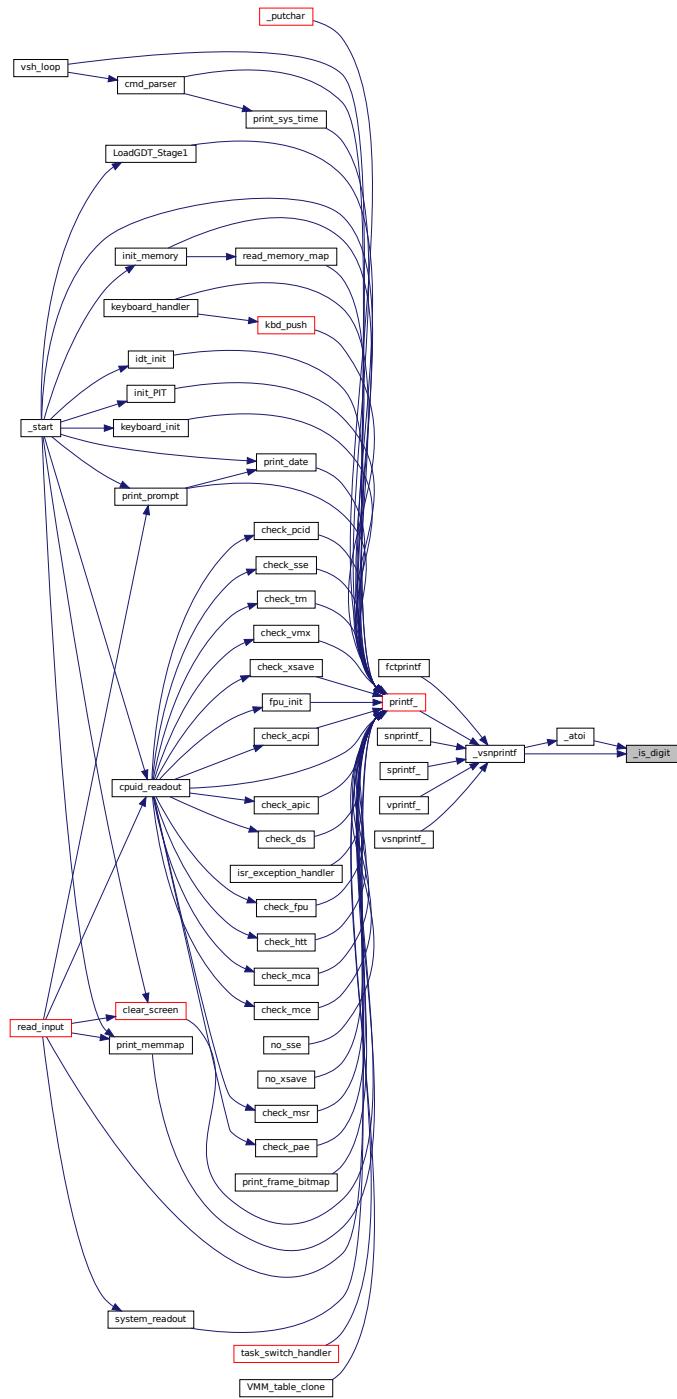


#### 4.117.3.4 \_is\_digit()

```
static bool _is_digit (
    char ch ) [inline], [static]
```

Definition at line 182 of file [printf.c](#).

Here is the caller graph for this function:



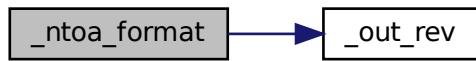
#### 4.117.3.5 \_ntoa\_format()

```
static size_t _ntoa_format (
    out_fct_type out,
```

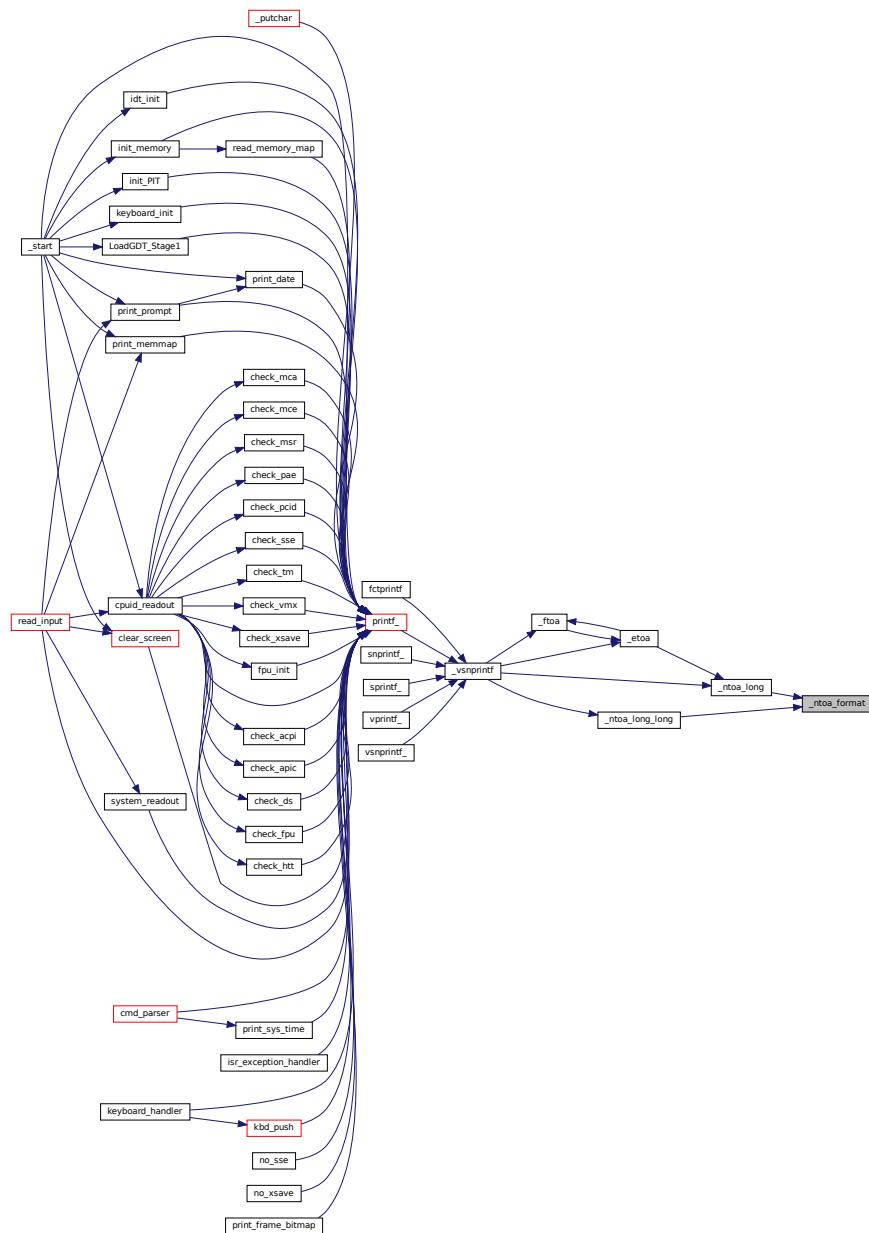
```
char * buffer,
size_t idx,
size_t maxlen,
char * buf,
size_t len,
bool negative,
unsigned int base,
unsigned int prec,
unsigned int width,
unsigned int flags ) [static]
```

Definition at line 231 of file [printf.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.117.3.6 \_ntoa\_long()

```

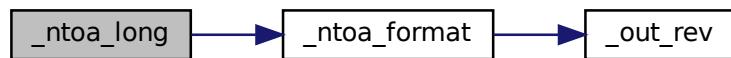
static size_t _ntoa_long (
    out_fct_type out,
    char * buffer,
    size_t idx,
    size_t maxlen,
    unsigned long value,
    bool negative,

```

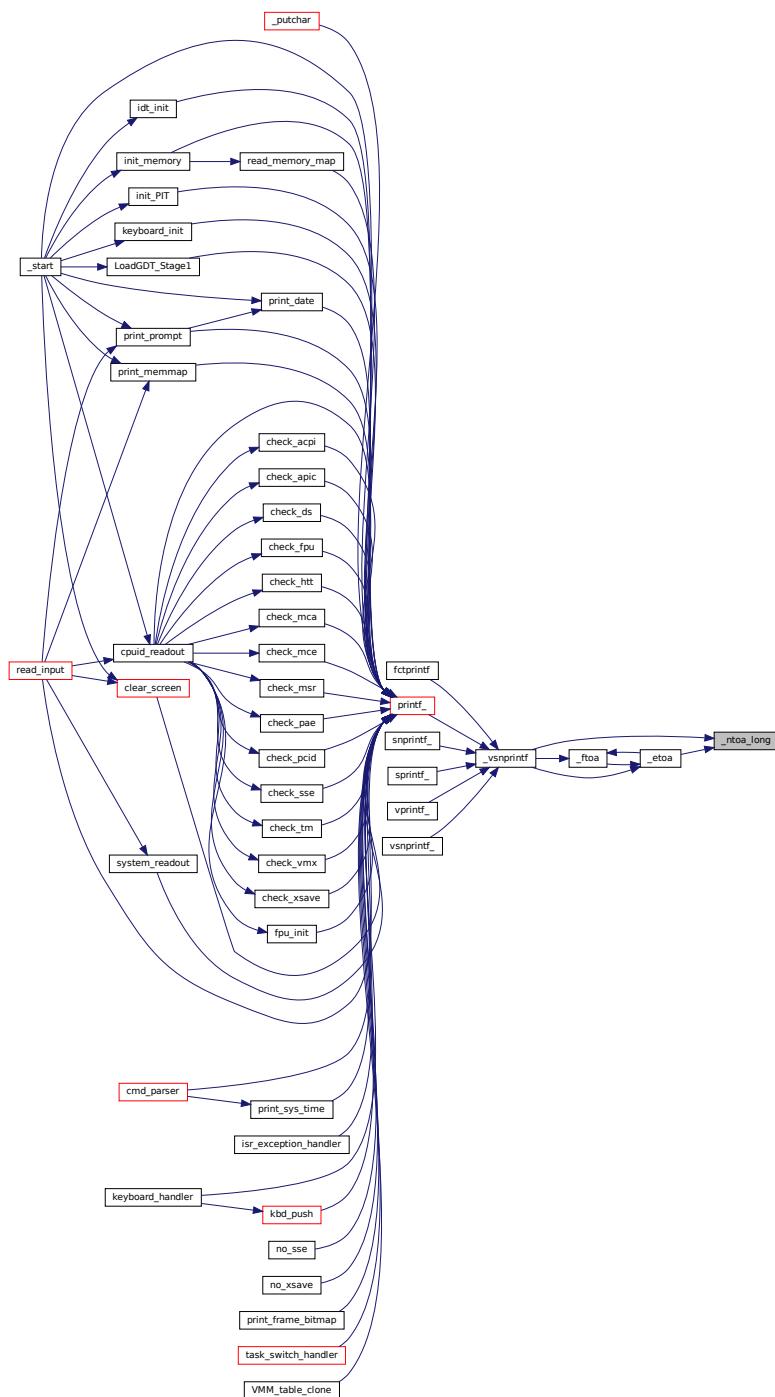
```
unsigned long base,  
unsigned int prec,  
unsigned int width,  
unsigned int flags ) [static]
```

Definition at line 299 of file [printf.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



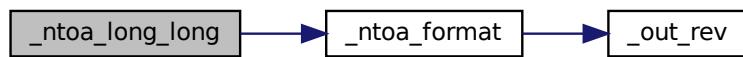
#### 4.117.3.7 `_ntoa_long()`

```
static size_t _ntoa_long_long (
    out_fct_type out,
```

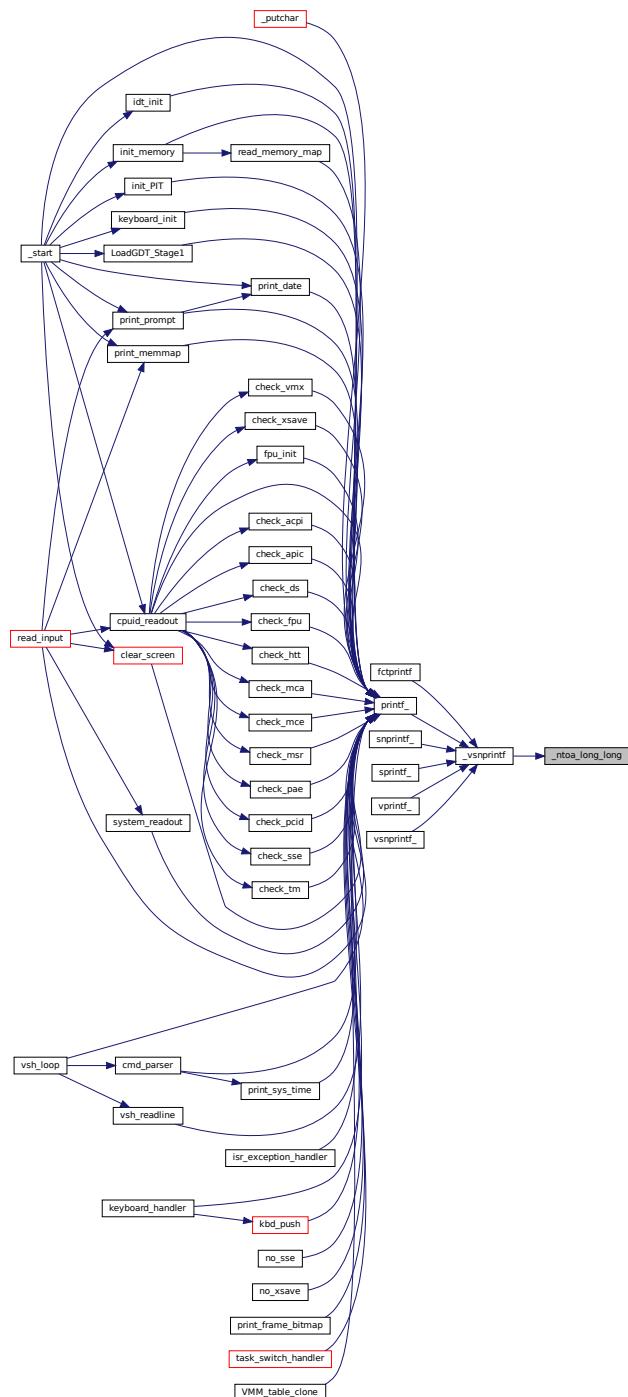
```
char * buffer,
size_t idx,
size_t maxlen,
unsigned long long value,
bool negative,
unsigned long long base,
unsigned int prec,
unsigned int width,
unsigned int flags ) [static]
```

Definition at line 326 of file [printf.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



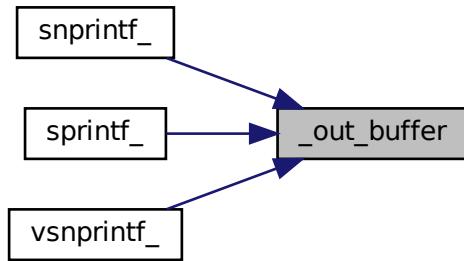
#### 4.117.3.8 `_out_buffer()`

```
static void _out_buffer (
    char character,
```

```
void * buffer,
size_t idx,
size_t maxlen ) [inline], [static]
```

Definition at line 129 of file [printf.c](#).

Here is the caller graph for this function:

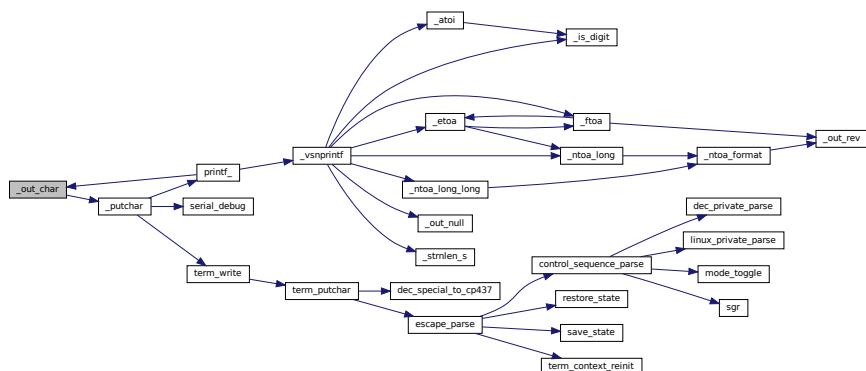


#### 4.117.3.9 \_out\_char()

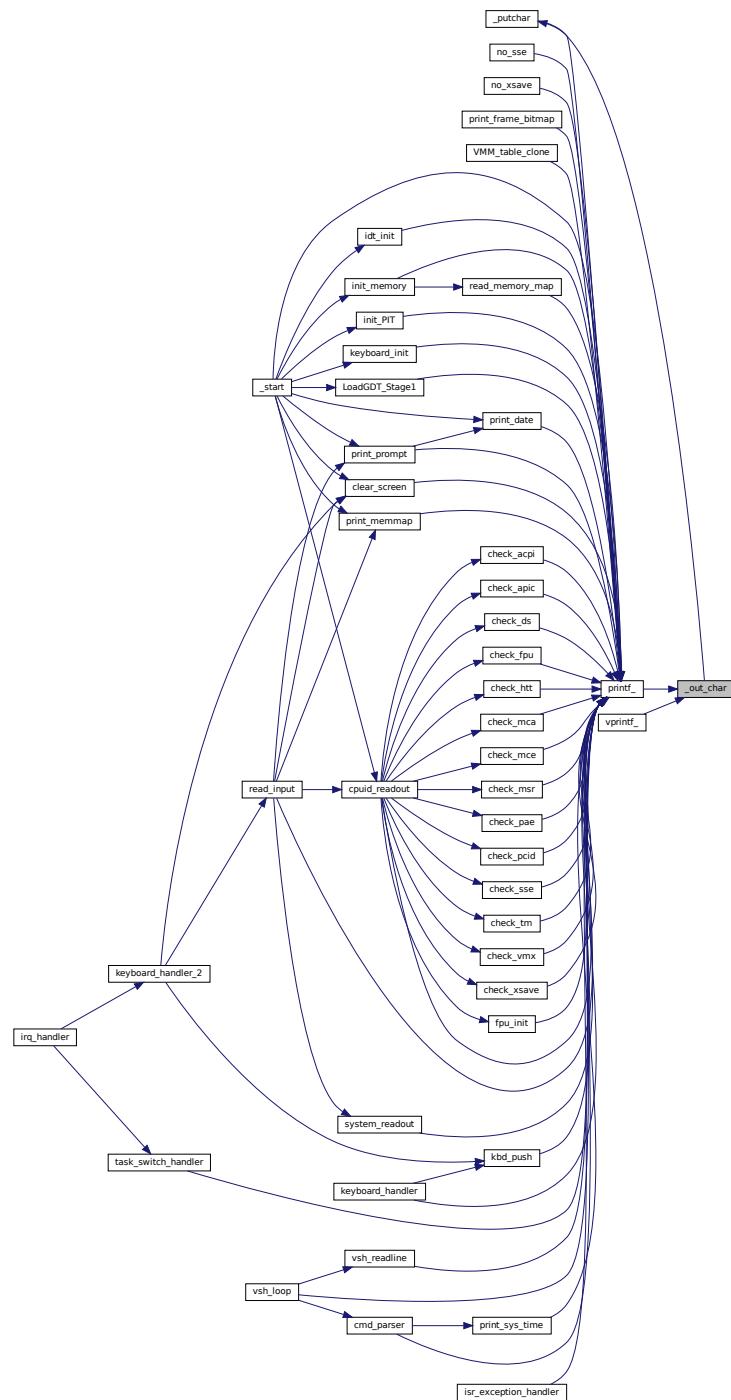
```
static void _out_char (
    char character,
    void * buffer,
    size_t idx,
    size_t maxlen ) [inline], [static]
```

Definition at line 147 of file [printf.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.117.3.10 `_out_fct()`

```
static void _out_fct (
    char character,
```

```
void * buffer,
size_t idx,
size_t maxlen ) [inline], [static]
```

Definition at line 159 of file [printf.c](#).

Here is the caller graph for this function:

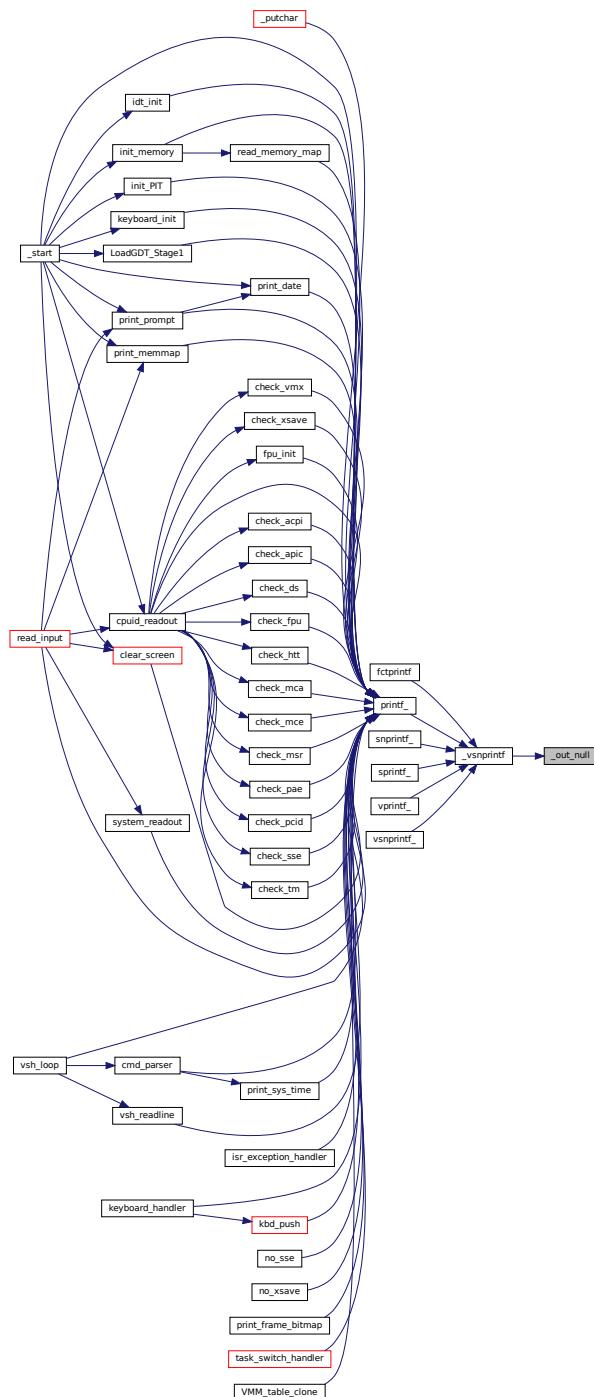


#### 4.117.3.11 \_out\_null()

```
static void _out_null (
    char character,
    void * buffer,
    size_t idx,
    size_t maxlen ) [inline], [static]
```

Definition at line 138 of file [printf.c](#).

Here is the caller graph for this function:



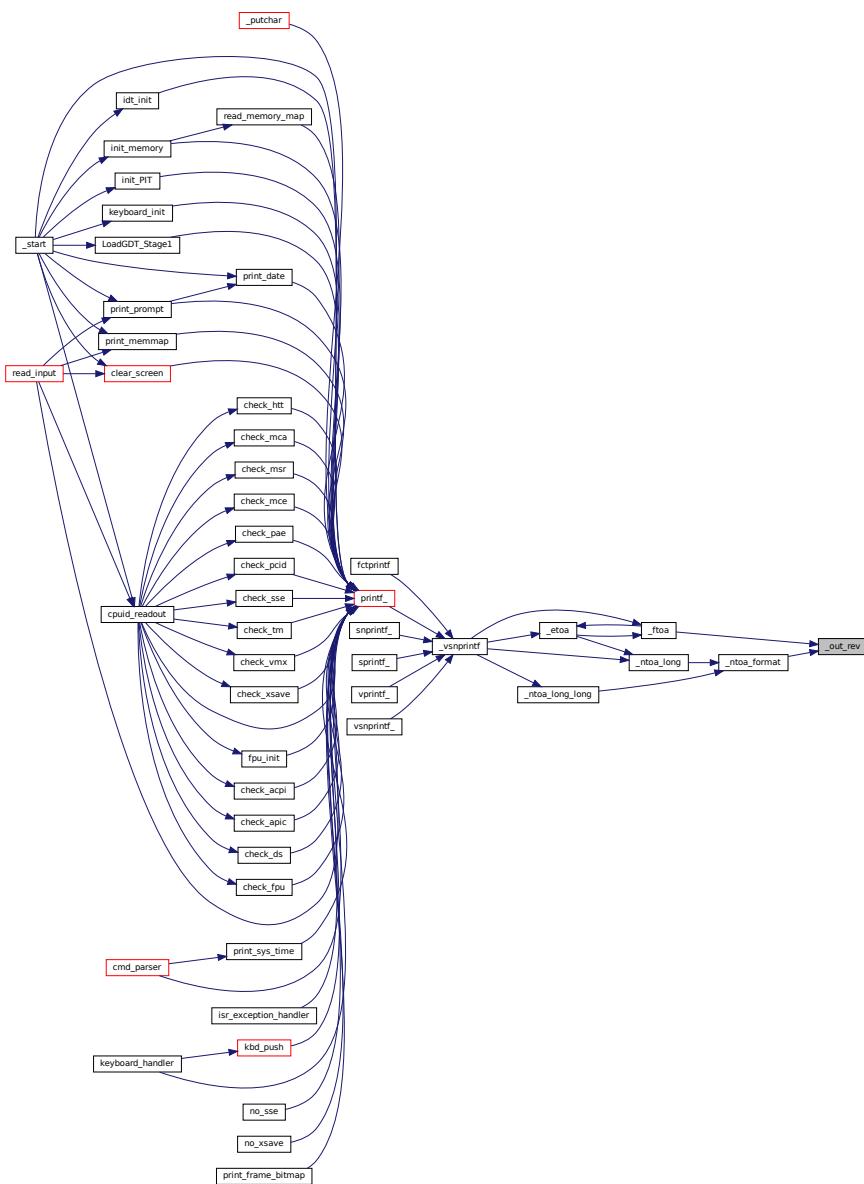
#### 4.117.3.12 `_out_rev()`

```
static size_t _out_rev (
    out_fct_type out,
```

```
char * buffer,  
size_t idx,  
size_t maxlen,  
const char * buf,  
size_t len,  
unsigned int width,  
unsigned int flags ) [static]
```

Definition at line 199 of file [printf.c](#).

Here is the caller graph for this function:

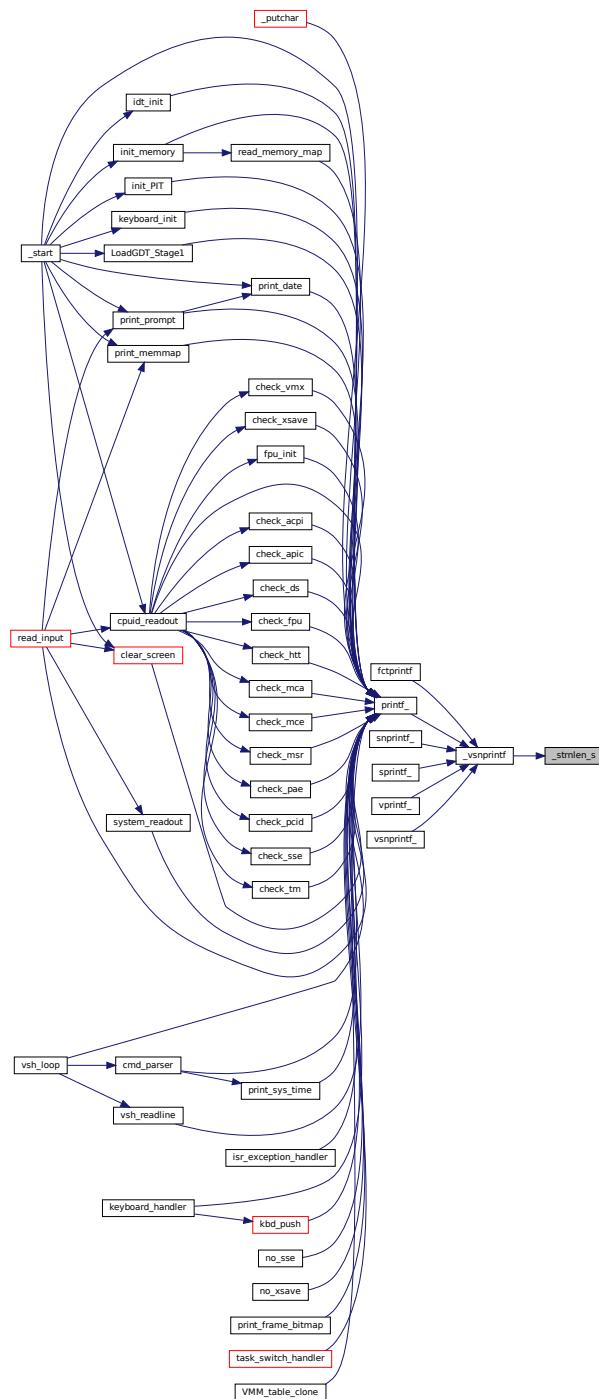


### 4.117.3.13 `_strnlen_s()`

```
static unsigned int _strnlen_s (
    const char * str,
    size_t maxsize ) [inline], [static]
```

Definition at line 172 of file [printf.c](#).

Here is the caller graph for this function:

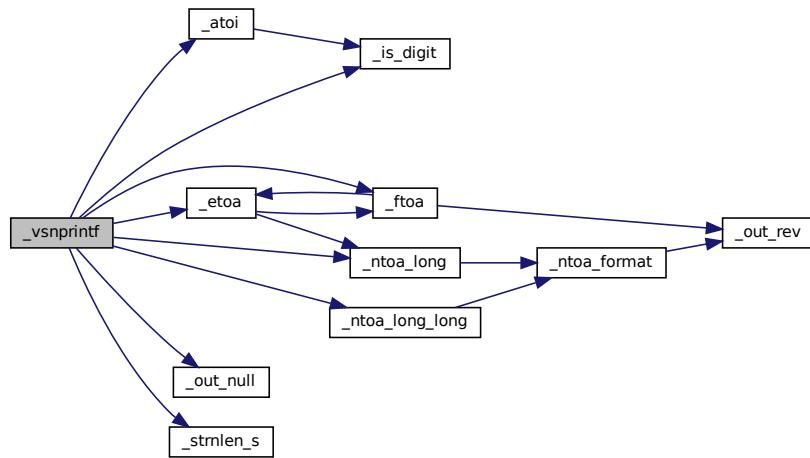


#### 4.117.3.14 `_vsnprintf()`

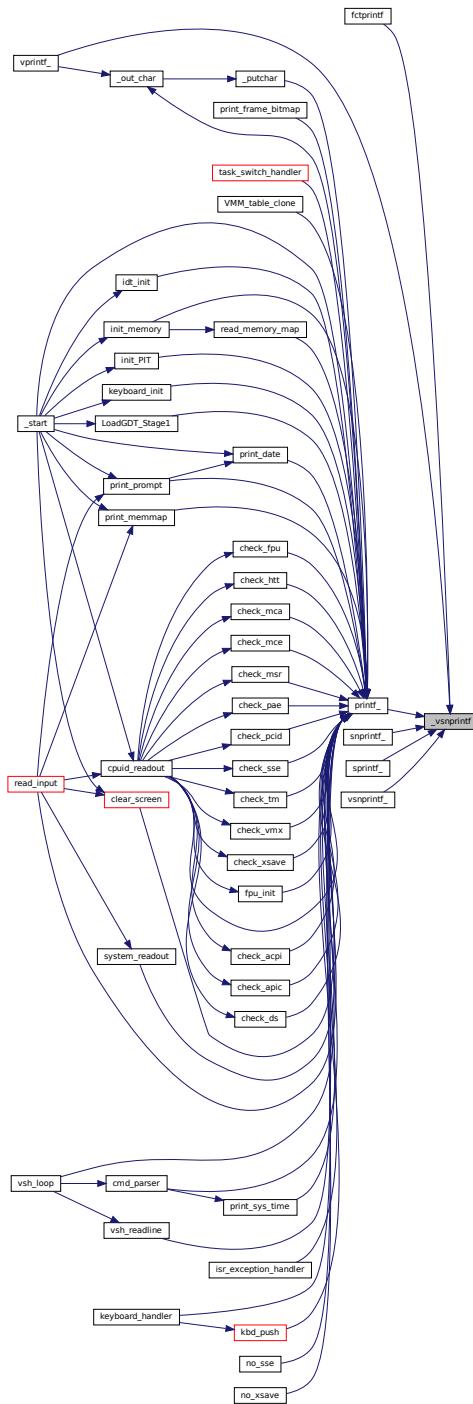
```
static int _vsnprintf (
    out_fct_type out,
    char * buffer,
    const size_t maxlen,
    const char * format,
    va_list va ) [static]
```

Definition at line 639 of file [printf.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.117.3.15 fctprintf()

```
int fctprintf (
    void(*)(char character, void *arg) out,
```

```
void * arg,
const char * format,
... )
```

printf with output function You may use this as dynamic alternative to [printf\(\)](#) with its fixed [\\_putchar\(\)](#) output

#### Parameters

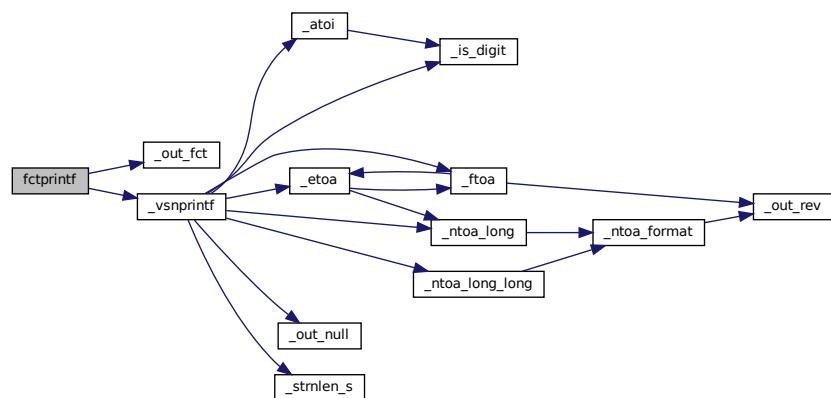
<i>out</i>	An output function which takes one character and an argument pointer
<i>arg</i>	An argument pointer for user data passed to output function
<i>format</i>	A string that specifies the format of the output

#### Returns

The number of characters that are sent to the output function, not counting the terminating null character

Definition at line 1036 of file [printf.c](#).

Here is the call graph for this function:

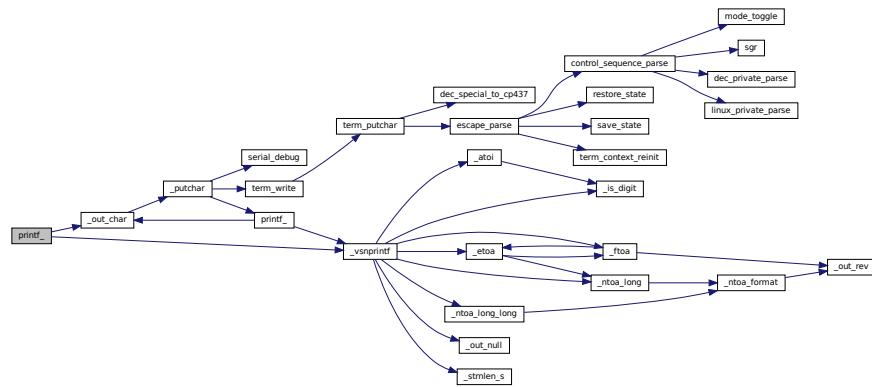


#### 4.117.3.16 printf\_()

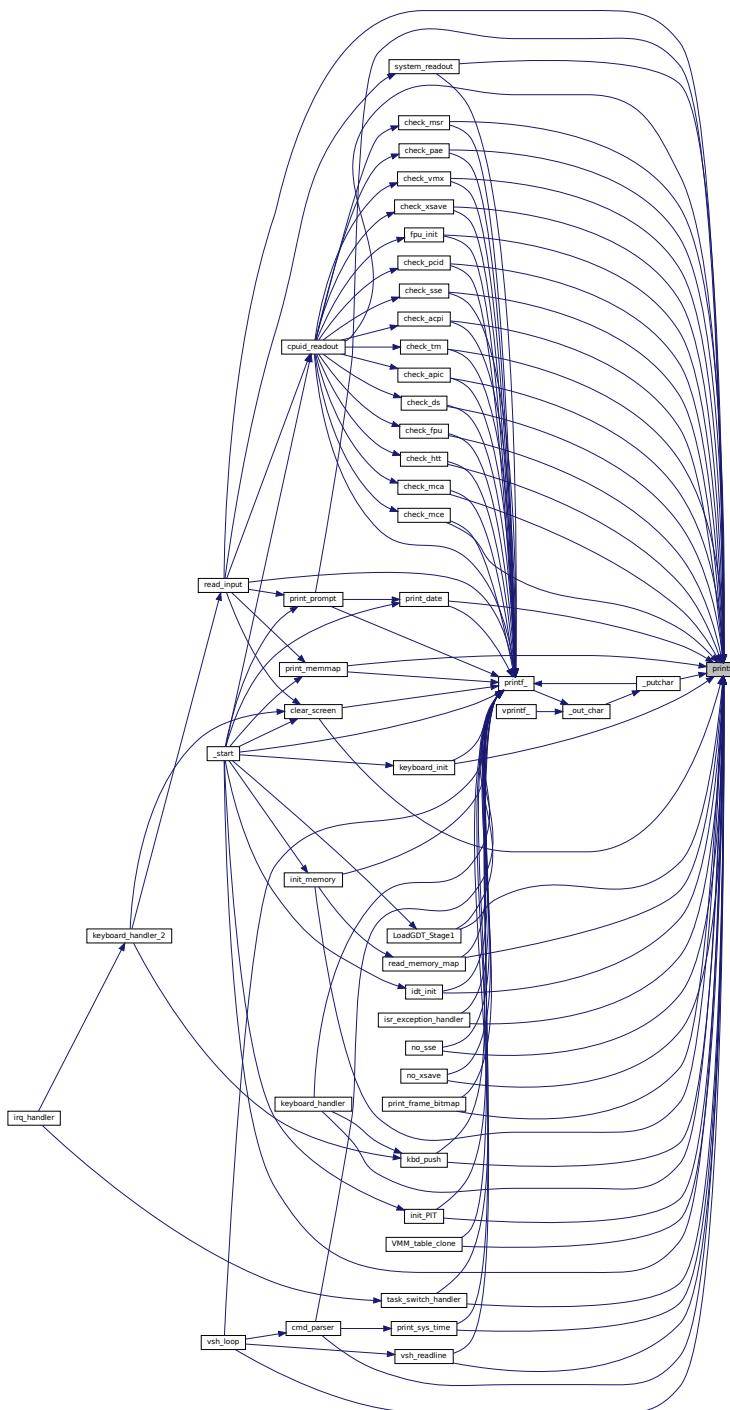
```
int printf_ (
    const char * format,
    ... )
```

Definition at line 997 of file [printf.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



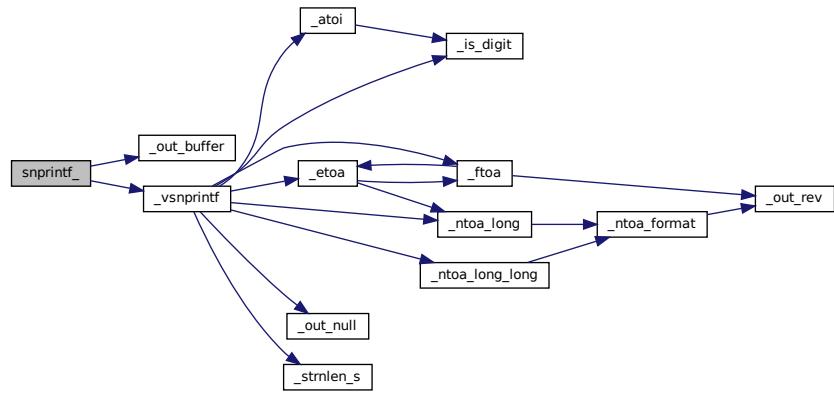
#### 4.117.3.17 snprintf\_()

```
int snprintf_(
    char * buffer,
```

```
size_t count,
const char * format,
... )
```

Definition at line 1016 of file [printf.c](#).

Here is the call graph for this function:

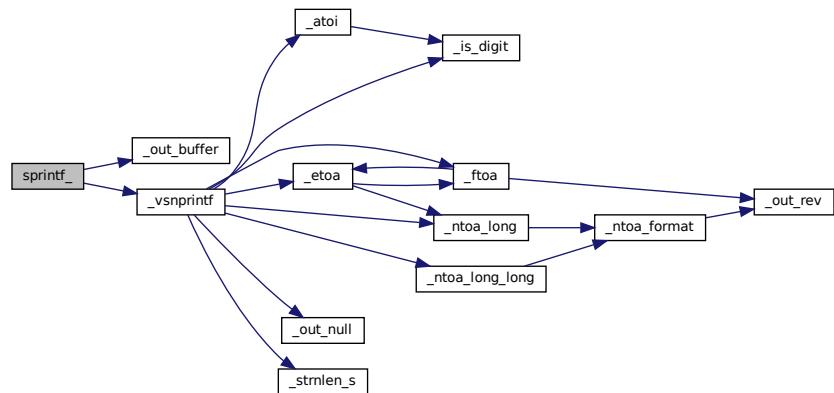


#### 4.117.3.18 sprintf\_()

```
int sprintf_ (
    char * buffer,
    const char * format,
    ... )
```

Definition at line 1007 of file [printf.c](#).

Here is the call graph for this function:

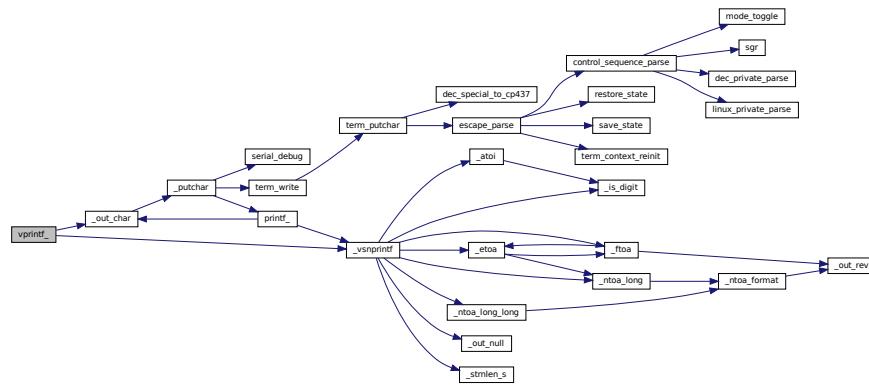


#### 4.117.3.19 vprintf\_()

```
int vprintf_ (
    const char * format,
    va_list va )
```

Definition at line 1025 of file [printf.c](#).

Here is the call graph for this function:

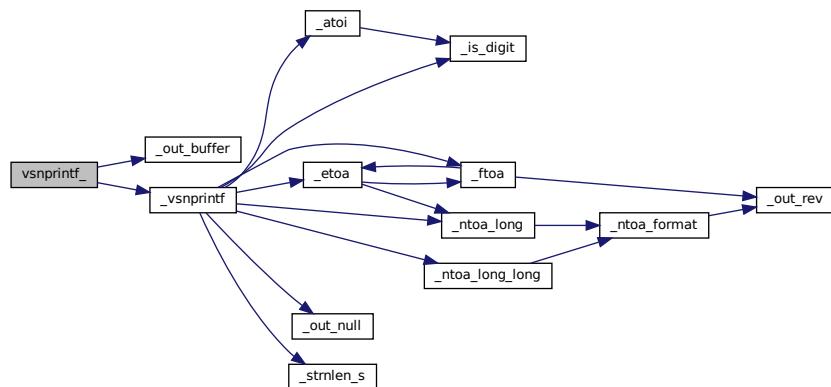


#### 4.117.3.20 vsnprintf\_()

```
int vsnprintf_ (
    char * buffer,
    size_t count,
    const char * format,
    va_list va )
```

Definition at line 1031 of file [printf.c](#).

Here is the call graph for this function:



## 4.118 printf.c

```
00001 // \author (c) Marco Paland (info@paland.com)
00003 //           2014-2019, PALANDesign Hannover, Germany
00004 //
00005 // \license The MIT License (MIT)
00006 //
00007 // Permission is hereby granted, free of charge, to any person obtaining a copy
00008 // of this software and associated documentation files (the "Software"), to deal
00009 // in the Software without restriction, including without limitation the rights
00010 // to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00011 // copies of the Software, and to permit persons to whom the Software is
00012 // furnished to do so, subject to the following conditions:
00013 //
00014 // The above copyright notice and this permission notice shall be included in
00015 // all copies or substantial portions of the Software.
00016 //
00017 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00018 // IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00019 // FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00020 // AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00021 // LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00022 // OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
00023 // THE SOFTWARE.
00024 //
00025 // \brief Tiny printf, sprintf and (v)snprintf implementation, optimized for speed on
00026 //       embedded systems with a very limited resources. These routines are thread
00027 //       safe and reentrant!
00028 //       Use this instead of the bloated standard/newlib printf cause these use
00029 //       malloc for printf (and may not be thread safe).
00030 //
00032
00033 #include <stdbool.h>
00034 #include <stdint.h>
00035
00036 #include "include/printf.h"
00037 #include "include/serial.h"
00038
00039 // define this globally (e.g. gcc -DPRINTF_INCLUDE_CONFIG_H ...) to include the
00040 // printf_config.h header file
00041 // default: undefined
00042 #ifdef PRINTF_INCLUDE_CONFIG_H
00043 #include "printf_config.h"
00044 #endif
00045
00046 // 'ntoa' conversion buffer size, this must be big enough to hold one converted
00047 // numeric number including padded zeros (dynamically created on stack)
00048 // default: 32 byte
00049 #ifndef PRINTF_NTOA_BUFFER_SIZE
00050 #define PRINTF_NTOA_BUFFER_SIZE 32U
00051 #endif
00052
00053 // 'ftoa' conversion buffer size, this must be big enough to hold one converted
00054 // float number including padded zeros (dynamically created on stack)
00055 // default: 32 byte
00056 #ifndef PRINTF_FTOA_BUFFER_SIZE
00057 #define PRINTF_FTOA_BUFFER_SIZE 32U
00058 #endif
00059
00060 // support for the floating point type (%f)
00061 // default: activated
00062 #ifndef PRINTF_DISABLE_SUPPORT_FLOAT
00063 #define PRINTF_SUPPORT_FLOAT
00064 #endif
00065
00066 // support for exponential floating point notation (%e/%g)
00067 // default: activated
00068 #ifndef PRINTF_DISABLE_SUPPORT_EXPONENTIAL
00069 #define PRINTF_SUPPORT_EXPONENTIAL
00070 #endif
00071
00072 // define the default floating point precision
00073 // default: 6 digits
00074 #ifndef PRINTF_DEFAULT_FLOAT_PRECISION
00075 #define PRINTF_DEFAULT_FLOAT_PRECISION 6U
00076 #endif
00077
00078 // define the largest float suitable to print with %f
00079 // default: 1e9
00080 #ifndef PRINTF_MAX_FLOAT
00081 #define PRINTF_MAX_FLOAT 1e9
00082 #endif
00083
00084 // support for the long long types (%llu or %p)
00085 // default: activated
00086 #ifndef PRINTF_DISABLE_SUPPORT_LONG_LONG
00087 #define PRINTF_SUPPORT_LONG_LONG
```

```

00088 #endif
00089
00090 // support for the ptrdiff_t type (%t)
00091 // ptrdiff_t is normally defined in <stddef.h> as long or long long type
00092 // default: activated
00093 #ifndef PRINTF_DISABLE_SUPPORT_PTRDIFF_T
00094 #define PRINTF_SUPPORT_PTRDIFF_T
00095 #endif
00096
00098
00099 // internal flag definitions
00100 #define FLAGS_ZEROPAD (1U « 0U)
00101 #define FLAGS_LEFT (1U « 1U)
00102 #define FLAGS_PLUS (1U « 2U)
00103 #define FLAGS_SPACE (1U « 3U)
00104 #define FLAGS_HASH (1U « 4U)
00105 #define FLAGS_UPPERCASE (1U « 5U)
00106 #define FLAGS_CHAR (1U « 6U)
00107 #define FLAGS_SHORT (1U « 7U)
00108 #define FLAGS_LONG (1U « 8U)
00109 #define FLAGS_LONG_LONG (1U « 9U)
00110 #define FLAGS_PRECISION (1U « 10U)
00111 #define FLAGS_ADAPT_EXP (1U « 11U)
00112
00113 // import float.h for DBL_MAX
00114 #if defined(PRINTF_SUPPORT_FLOAT)
00115 #include <float.h>
00116 #endif
00117
00118 // output function type
00119 typedef void (*out_fct_type)(char character, void *buffer, size_t idx, size_t maxlen);
00120
00121 // wrapper (used as buffer) for output function type
00122 typedef struct
00123 {
00124     void (*fct)(char character, void *arg);
00125     void *arg;
00126 } out_fct_wrap_type;
00127
00128 // internal buffer output
00129 static inline void _out_buffer(char character, void *buffer, size_t idx, size_t maxlen)
00130 {
00131     if (idx < maxlen)
00132     {
00133         ((char *)buffer)[idx] = character;
00134     }
00135 }
00136
00137 // internal null output
00138 static inline void _out_null(char character, void *buffer, size_t idx, size_t maxlen)
00139 {
00140     (void)character;
00141     (void)buffer;
00142     (void)idx;
00143     (void)maxlen;
00144 }
00145
00146 // internal _putchar wrapper
00147 static inline void _out_char(char character, void *buffer, size_t idx, size_t maxlen)
00148 {
00149     (void)buffer;
00150     (void)idx;
00151     (void)maxlen;
00152     if (character)
00153     {
00154         _putchar(character);
00155     }
00156 }
00157
00158 // internal output function wrapper
00159 static inline void _out_fct(char character, void *buffer, size_t idx, size_t maxlen)
00160 {
00161     (void)idx;
00162     (void)maxlen;
00163     if (character)
00164     {
00165         // buffer is the output fct pointer
00166         ((out_fct_wrap_type *)buffer)->fct(character, ((out_fct_wrap_type *)buffer)->arg);
00167     }
00168 }
00169
00170 // internal secure strlen
00171 // \return The length of the string (excluding the terminating 0) limited by 'maxsize'
00172 static inline unsigned int _strnlen_s(const char *str, size_t maxsize)
00173 {
00174     const char *s;
00175     for (s = str; *s && maxsize--; ++s)

```

```

00176      ;
00177      return (unsigned int)(s - str);
00178 }
00179
00180 // internal test if char is a digit (0-9)
00181 // \return true if char is a digit
00182 static inline bool _is_digit(char ch)
00183 {
00184     return (ch >= '0') && (ch <= '9');
00185 }
00186
00187 // internal ASCII string to unsigned int conversion
00188 static unsigned int _atoi(const char **str)
00189 {
00190     unsigned int i = 0U;
00191     while (_is_digit(**str))
00192     {
00193         i = i * 10U + (unsigned int)(*((*str)++) - '0');
00194     }
00195     return i;
00196 }
00197
00198 // output the specified string in reverse, taking care of any zero-padding
00199 static size_t _out_rev(out_fct_type out, char *buffer, size_t idx, size_t maxlen, const char *buf,
00200 size_t len, unsigned int width, unsigned int flags)
00201 {
00202     const size_t start_idx = idx;
00203
00204     // pad spaces up to given width
00205     if (!(flags & FLAGS_LEFT) && !(flags & FLAGS_ZEROPAD))
00206     {
00207         for (size_t i = len; i < width; i++)
00208         {
00209             out(' ', buffer, idx++, maxlen);
00210         }
00211     }
00212
00213     // reverse string
00214     while (len)
00215     {
00216         out(buf[--len], buffer, idx++, maxlen);
00217     }
00218
00219     // append pad spaces up to given width
00220     if (flags & FLAGS_LEFT)
00221     {
00222         while (idx - start_idx < width)
00223         {
00224             out(' ', buffer, idx++, maxlen);
00225         }
00226     }
00227     return idx;
00228 }
00229
00230 // internal itoa format
00231 static size_t _ntoa_format(out_fct_type out, char *buffer, size_t idx, size_t maxlen, char *buf,
00232 size_t len, bool negative, unsigned int base, unsigned int prec, unsigned int width, unsigned int
00233 flags)
00234 {
00235     // pad leading zeros
00236     if (!(flags & FLAGS_LEFT))
00237     {
00238         if (width && (flags & FLAGS_ZEROPAD) && (negative || (flags & (FLAGS_PLUS | FLAGS_SPACE))))
00239         {
00240             width--;
00241         }
00242         while ((len < prec) && (len < PRINTF_NTOA_BUFFER_SIZE))
00243         {
00244             buf[len++] = '0';
00245         }
00246         while ((flags & FLAGS_ZEROPAD) && (len < width) && (len < PRINTF_NTOA_BUFFER_SIZE))
00247         {
00248             buf[len++] = '0';
00249         }
00250
00251     // handle hash
00252     if (flags & FLAGS_HASH)
00253     {
00254         if (!(flags & FLAGS_PRECISION) && len && ((len == prec) || (len == width)))
00255         {
00256             len--;
00257             if (len && (base == 16U))
00258             {
00259                 len--;
00260             }
00261         }
00262     }
00263 }

```

```

00260         }
00261         if ((base == 16U) && !(flags & FLAGS_UPPERCASE) && (len < PRINTF_NTOA_BUFFER_SIZE))
00262     {
00263         buf[len++] = 'x';
00264     }
00265     else if ((base == 16U) && (flags & FLAGS_UPPERCASE) && (len < PRINTF_NTOA_BUFFER_SIZE))
00266     {
00267         buf[len++] = 'X';
00268     }
00269     else if ((base == 2U) && (len < PRINTF_NTOA_BUFFER_SIZE))
00270     {
00271         buf[len++] = 'b';
00272     }
00273     if (len < PRINTF_NTOA_BUFFER_SIZE)
00274     {
00275         buf[len++] = '0';
00276     }
00277 }
00278
00279 if (len < PRINTF_NTOA_BUFFER_SIZE)
00280 {
00281     if (negative)
00282     {
00283         buf[len++] = '-';
00284     }
00285     else if (flags & FLAGS_PLUS)
00286     {
00287         buf[len++] = '+'; // ignore the space if the '+' exists
00288     }
00289     else if (flags & FLAGS_SPACE)
00290     {
00291         buf[len++] = ' ';
00292     }
00293 }
00294
00295 return _out_rev(out, buffer, idx, maxlen, buf, len, width, flags);
00296 }
00297
00298 // internal itoa for 'long' type
00299 static size_t _ntoa_long(out_fct_type out, char *buffer, size_t idx, size_t maxlen, unsigned long
value, bool negative, unsigned long base, unsigned int prec, unsigned int width, unsigned int flags)
00300 {
00301     char buf[PRINTF_NTOA_BUFFER_SIZE];
00302     size_t len = 0U;
00303
00304     // no hash for 0 values
00305     if (!value)
00306     {
00307         flags &= ~FLAGS_HASH;
00308     }
00309
00310     // write if precision != 0 and value is != 0
00311     if (!(flags & FLAGS_PRECISION) || value)
00312     {
00313         do
00314         {
00315             const char digit = (char)(value % base);
00316             buf[len++] = digit < 10 ? '0' + digit : (flags & FLAGS_UPPERCASE ? 'A' : 'a') + digit -
10;
00317             value /= base;
00318         } while (value && (len < PRINTF_NTOA_BUFFER_SIZE));
00319     }
00320
00321     return _ntoa_format(out, buffer, idx, maxlen, buf, len, negative, (unsigned int)base, prec, width,
flags);
00322 }
00323
00324 // internal itoa for 'long long' type
00325 #if defined(PRINTF_SUPPORT_LONG_LONG)
00326 static size_t _ntoa_long_long(out_fct_type out, char *buffer, size_t idx, size_t maxlen, unsigned long
long value, bool negative, unsigned long long base, unsigned int prec, unsigned int width, unsigned
int flags)
00327 {
00328     char buf[PRINTF_NTOA_BUFFER_SIZE];
00329     size_t len = 0U;
00330
00331     // no hash for 0 values
00332     if (!value)
00333     {
00334         flags &= ~FLAGS_HASH;
00335     }
00336
00337     // write if precision != 0 and value is != 0
00338     if (!(flags & FLAGS_PRECISION) || value)
00339     {
00340         do
00341         {

```

```

00342         const char digit = (char)(value % base);
00343         buf[len++] = digit < 10 ? '0' + digit : (flags & FLAGS_UPPERCASE ? 'A' : 'a') + digit -
00344             value /= base;
00345     } while (value && (len < PRINTF_NTOA_BUFFER_SIZE));
00346 }
00347
00348     return _ntoa_format(out, buffer, idx, maxlen, buf, len, negative, (unsigned int)base, prec, width,
00349     flags);
00350 #endif // PRINTF_SUPPORT_LONG_LONG
00351
00352 #if defined(PRINTF_SUPPORT_FLOAT)
00353
00354 #if defined(PRINTF_SUPPORT_EXPONENTIAL)
00355 // forward declaration so that _ftoa can switch to exp notation for values > PRINTF_MAX_FLOAT
00356 static size_t _ftoa(out_fct_type out, char *buffer, size_t idx, size_t maxlen, double value, unsigned
00357     int prec, unsigned int width, unsigned int flags);
00358 #endif
00359 // internal ftoa for fixed decimal floating point
00360 static size_t _ftoa(out_fct_type out, char *buffer, size_t idx, size_t maxlen, double value, unsigned
00361     int prec, unsigned int width, unsigned int flags)
00362 {
00363     char buf[PRINTF_FTOA_BUFFER_SIZE];
00364     size_t len = 0U;
00365     double diff = 0.0;
00366
00367     // powers of 10
00368     static const double pow10[] = {1, 10, 100, 1000, 10000, 100000, 1000000, 10000000, 100000000,
00369     1000000000};
00370
00371     // test for special values
00372     if (value != value)
00373         return _out_rev(out, buffer, idx, maxlen, "nan", 3, width, flags);
00374     if (value < -DBL_MAX)
00375         return _out_rev(out, buffer, idx, maxlen, "fni-", 4, width, flags);
00376     if (value > DBL_MAX)
00377         return _out_rev(out, buffer, idx, maxlen, (flags & FLAGS_PLUS) ? "fni+" : "fni", (flags &
00378         FLAGS_PLUS) ? 4U : 3U, width, flags);
00379
00380     // test for very large values
00381     // standard printf behavior is to print EVERY whole number digit -- which could be 100s of
00382     // characters overflowing your buffers == bad
00383     if ((value > PRINTF_MAX_FLOAT) || (value < -PRINTF_MAX_FLOAT))
00384     {
00385 #if defined(PRINTF_SUPPORT_EXPONENTIAL)
00386         return _ftoa(out, buffer, idx, maxlen, value, prec, width, flags);
00387 #else
00388         return 0U;
00389 #endif
00390     }
00391
00392     // test for negative
00393     bool negative = false;
00394     if (value < 0)
00395     {
00396         negative = true;
00397         value = 0 - value;
00398     }
00399
00400     // set default precision, if not set explicitly
00401     if (!(flags & FLAGS_PRECISION))
00402     {
00403         prec = PRINTF_DEFAULT_FLOAT_PRECISION;
00404     }
00405     // limit precision to 9, cause a prec >= 10 can lead to overflow errors
00406     while ((len < PRINTF_FTOA_BUFFER_SIZE) && (prec > 9U))
00407     {
00408         buf[len++] = '0';
00409         prec--;
00410     }
00411
00412     int whole = (int)value;
00413     double tmp = (value - whole) * pow10[prec];
00414     unsigned long frac = (unsigned long)tmp;
00415     diff = tmp - frac;
00416
00417     if (diff > 0.5)
00418     {
00419         ++frac;
00420         // handle rollover, e.g. case 0.99 with prec 1 is 1.0
00421         if (frac >= pow10[prec])
00422         {
00423             frac = 0;
00424             ++whole;
00425         }
00426     }

```

```

00422     }
00423     else if (diff < 0.5)
00424     {
00425     }
00426     else if ((frac == 0U) || (frac & 1U))
00427     {
00428         // if halfway, round up if odd OR if last digit is 0
00429         ++frac;
00430     }
00431
00432     if (prec == 0U)
00433     {
00434         diff = value - (double)whole;
00435         if ((!(diff < 0.5) || (diff > 0.5)) && (whole & 1))
00436         {
00437             // exactly 0.5 and ODD, then round up
00438             // 1.5 -> 2, but 2.5 -> 2
00439             ++whole;
00440         }
00441     }
00442     else
00443     {
00444         unsigned int count = prec;
00445         // now do fractional part, as an unsigned number
00446         while (len < PRINTF_FTOA_BUFFER_SIZE)
00447         {
00448             --count;
00449             buf[len++] = (char)(48U + (frac % 10U));
00450             if (!(frac /= 10U))
00451             {
00452                 break;
00453             }
00454         }
00455         // add extra 0s
00456         while ((len < PRINTF_FTOA_BUFFER_SIZE) && (count-- > 0U))
00457         {
00458             buf[len++] = '0';
00459         }
00460         if (len < PRINTF_FTOA_BUFFER_SIZE)
00461         {
00462             // add decimal
00463             buf[len++] = '.';
00464         }
00465     }
00466
00467     // do whole part, number is reversed
00468     while (len < PRINTF_FTOA_BUFFER_SIZE)
00469     {
00470         buf[len++] = (char)(48 + (whole % 10));
00471         if (!(whole /= 10))
00472         {
00473             break;
00474         }
00475     }
00476
00477     // pad leading zeros
00478     if (!(flags & FLAGS_LEFT) && (flags & FLAGS_ZEROPAD))
00479     {
00480         if (width && (negative || (flags & (FLAGS_PLUS | FLAGS_SPACE))))
00481         {
00482             width--;
00483         }
00484         while ((len < width) && (len < PRINTF_FTOA_BUFFER_SIZE))
00485         {
00486             buf[len++] = '0';
00487         }
00488     }
00489
00490     if (len < PRINTF_FTOA_BUFFER_SIZE)
00491     {
00492         if (negative)
00493         {
00494             buf[len++] = '-';
00495         }
00496         else if (flags & FLAGS_PLUS)
00497         {
00498             buf[len++] = '+'; // ignore the space if the '+' exists
00499         }
00500         else if (flags & FLAGS_SPACE)
00501         {
00502             buf[len++] = ' ';
00503         }
00504     }
00505
00506     return _out_rev(out, buffer, idx, maxlen, buf, len, width, flags);
00507 }
00508

```

```

00509 //if defined(PRINTF_SUPPORT_EXPONENTIAL)
00510 // internal ftoa variant for exponential floating-point type, contributed by Martijn Jasperse
00511 <m.jasperse@gmail.com>
00512 static size_t _etoa(out_fct_type out, char *buffer, size_t idx, size_t maxlen, double value, unsigned
00513 int prec, unsigned int width, unsigned int flags)
00514 {
00515     // check for NaN and special values
00516     if ((value != value) || (value > DBL_MAX) || (value < -DBL_MAX))
00517     {
00518         return _ftoa(out, buffer, idx, maxlen, value, prec, width, flags);
00519     }
00520 
00521     // determine the sign
00522     const bool negative = value < 0;
00523     if (negative)
00524     {
00525         value = -value;
00526     }
00527 
00528     // default precision
00529     if (!(flags & FLAGS_PRECISION))
00530     {
00531         prec = PRINTF_DEFAULT_FLOAT_PRECISION;
00532     }
00533 
00534     // determine the decimal exponent
00535     // based on the algorithm by David Gay (https://www.ampl.com/netlib/fp/dtoa.c)
00536     union
00537     {
00538         uint64_t U;
00539         double F;
00540     } conv;
00541 
00542     conv.F = value;
00543     int exp2 = (int)((conv.U >> 52U) & 0x07FFU) - 1023;           // effectively log2
00544     conv.U = (conv.U & ((1ULL << 52U) - 1U)) | (1023ULL << 52U); // drop the exponent so conv.F is now
00545     in [1,2]
00546     // now approximate log10 from the log2 integer part and an expansion of ln around 1.5
00547     int expval = (int)(0.1760912590558 + exp2 * 0.301029995663981 + (conv.F - 1.5) *
00548     0.289529654602168);
00549     // now we want to compute 10^expval but we want to be sure it won't overflow
00550     exp2 = (int)(expval * 3.321928094887362 + 0.5);
00551     const double z = expval * 2.302585092994046 - exp2 * 0.6931471805599453;
00552     const double z2 = z * z;
00553     conv.U = (uint64_t)(exp2 + 1023) << 52U;
00554     // compute exp(z) using continued fractions, see
00555     // https://en.wikipedia.org/wiki/Exponential\_function#Continued\_fractions\_for\_ex
00556     conv.F *= 1 + 2 * z / (2 - z + (z2 / (6 + (z2 / (10 + z2 / 14))))));
00557     // correct for rounding errors
00558     if (value < conv.F)
00559     {
00560         expval--;
00561         conv.F /= 10;
00562     }
00563 
00564     // the exponent format is "%+03d" and largest value is "307", so set aside 4-5 characters
00565     unsigned int minwidth = ((expval < 100) && (expval > -100)) ? 4U : 5U;
00566 
00567     // in "%g" mode, "prec" is the number of *significant figures* not decimals
00568     if (flags & FLAGS_ADAPT_EXP)
00569     {
00570         // do we want to fall-back to "%f" mode?
00571         if ((value >= 1e-4) && (value < 1e6))
00572         {
00573             if (((int)prec > expval)
00574                 {
00575                     prec = (unsigned)((int)prec - expval - 1);
00576                 }
00577             else
00578             {
00579                 prec = 0;
00580             }
00581         }
00582         flags |= FLAGS_PRECISION; // make sure _ftoa respects precision
00583         // no characters in exponent
00584         minwidth = 0U;
00585         expval = 0;
00586     }
00587     else
00588     {
00589         // we use one sigfig for the whole part
00590         if ((prec > 0) && (flags & FLAGS_PRECISION))
00591         {
00592             --prec;
00593         }
00594     }
00595 }
00596

```

```

00591 // will everything fit?
00592 unsigned int fwidth = width;
00593 if (width > minwidth)
00594 {
00595     // we didn't fall-back so subtract the characters required for the exponent
00596     fwidth -= minwidth;
00597 }
00598 else
00599 {
00600     // not enough characters, so go back to default sizing
00601     fwidth = 0U;
00602 }
00603 if ((flags & FLAGS_LEFT) && minwidth)
00604 {
00605     // if we're padding on the right, DON'T pad the floating part
00606     fwidth = 0U;
00607 }
00608
00609 // rescale the float value
00610 if (expval)
00611 {
00612     value /= conv.F;
00613 }
00614
00615 // output the floating part
00616 const size_t start_idx = idx;
00617 idx = _ftoa(out, buffer, idx, maxlen, negative ? -value : value, prec, fwidth, flags &
00618 ~FLAGS_ADAPT_EXP);
00619
00620 // output the exponent part
00621 if (minwidth)
00622 {
00623     // output the exponential symbol
00624     out((flags & FLAGS_UPPERCASE) ? 'E' : 'e', buffer, idx++, maxlen);
00625     // output the exponent value
00626     idx = _ntoa_long(out, buffer, idx, maxlen, (expval < 0) ? -expval : expval, expval < 0, 10, 0,
00627     minwidth - 1, FLAGS_ZEROPAD | FLAGS_PLUS);
00628     // might need to right-pad spaces
00629     if (flags & FLAGS_LEFT)
00630     {
00631         while (idx - start_idx < width)
00632             out(' ', buffer, idx++, maxlen);
00633     }
00634 }
00635 #endif // PRINTF_SUPPORT_EXPONENTIAL
00636 #endif // PRINTF_SUPPORT_FLOAT
00637
00638 // internal vsnprintf
00639 static int _vsnprintf(out_fct_type out, char *buffer, const size_t maxlen, const char *format, va_list
00640 va)
00641 {
00642     unsigned int flags, width, precision, n;
00643     size_t idx = 0U;
00644
00645     if (!buffer)
00646     {
00647         // use null output function
00648         out = _out_null;
00649     }
00650
00651     while (*format)
00652     {
00653         // format specifier? %[flags][width][.precision][length]
00654         if (*format != '%')
00655         {
00656             // no
00657             out(*format, buffer, idx++, maxlen);
00658             format++;
00659             continue;
00660         }
00661         else
00662         {
00663             // yes, evaluate it
00664             format++;
00665
00666             // evaluate flags
00667             flags = 0U;
00668             do
00669             {
00670                 switch (*format)
00671                 {
00672                     case '0':
00673                         flags |= FLAGS_ZEROPAD;
00674                         format++;

```

```
00675         n = 1U;
00676         break;
00677     case '-':
00678         flags |= FLAGS_LEFT;
00679         format++;
00680         n = 1U;
00681         break;
00682     case '+':
00683         flags |= FLAGS_PLUS;
00684         format++;
00685         n = 1U;
00686         break;
00687     case ' ':
00688         flags |= FLAGS_SPACE;
00689         format++;
00690         n = 1U;
00691         break;
00692     case '#':
00693         flags |= FLAGS_HASH;
00694         format++;
00695         n = 1U;
00696         break;
00697     default:
00698         n = 0U;
00699         break;
00700     }
00701 } while (n);

00702 // evaluate width field
00703 width = 0U;
00704 if (_is_digit(*format))
00705 {
00706     width = _atoi(&format);
00707 }
00708 else if (*format == '*')
00709 {
00710     const int w = va_arg(va, int);
00711     if (w < 0)
00712     {
00713         flags |= FLAGS_LEFT; // reverse padding
00714         width = (unsigned int)-w;
00715     }
00716     else
00717     {
00718         width = (unsigned int)w;
00719     }
00720     format++;
00721 }
00722 }

00723 // evaluate precision field
00724 precision = 0U;
00725 if (*format == '.')
00726 {
00727     flags |= FLAGS_PRECISION;
00728     format++;
00729     if (_is_digit(*format))
00730     {
00731         precision = _atoi(&format);
00732     }
00733     else if (*format == '*')
00734     {
00735         const int prec = (int)va_arg(va, int);
00736         precision = prec > 0 ? (unsigned int)prec : 0U;
00737         format++;
00738     }
00739 }
00740 }

00741 // evaluate length field
00742 switch (*format)
00743 {
00744     case 'l':
00745         flags |= FLAGS_LONG;
00746         format++;
00747         if (*format == 'l')
00748         {
00749             flags |= FLAGS_LONG_LONG;
00750             format++;
00751         }
00752         break;
00753     case 'h':
00754         flags |= FLAGS_SHORT;
00755         format++;
00756         if (*format == 'h')
00757         {
00758             flags |= FLAGS_CHAR;
00759             format++;
00760         }
00761 }
```

```

00762         break;
00763 #if defined(PRINTF_SUPPORT_PTRDIFF_T)
00764     case 't':
00765         flags |= (sizeof(ptrdiff_t) == sizeof(long) ? FLAGS_LONG : FLAGS_LONG_LONG);
00766         format++;
00767         break;
00768 #endif
00769     case 'j':
00770         flags |= (sizeof(intmax_t) == sizeof(long) ? FLAGS_LONG : FLAGS_LONG_LONG);
00771         format++;
00772         break;
00773     case 'z':
00774         flags |= (sizeof(size_t) == sizeof(long) ? FLAGS_LONG : FLAGS_LONG_LONG);
00775         format++;
00776         break;
00777     default:
00778         break;
00779     }
00780
00781 // evaluate specifier
00782 switch (*format)
00783 {
00784     case 'd':
00785     case 'i':
00786     case 'u':
00787     case 'x':
00788     case 'X':
00789     case 'o':
00790     case 'b':
00791     {
00792         // set the base
00793         unsigned int base;
00794         if (*format == 'x' || *format == 'X')
00795         {
00796             base = 16U;
00797         }
00798         else if (*format == 'o')
00799         {
00800             base = 8U;
00801         }
00802         else if (*format == 'b')
00803         {
00804             base = 2U;
00805         }
00806     else
00807     {
00808         base = 10U;
00809         flags &= ~FLAGS_HASH; // no hash for dec format
00810     }
00811     // uppercase
00812     if (*format == 'X')
00813     {
00814         flags |= FLAGS_UPPERCASE;
00815     }
00816
00817     // no plus or space flag for u, x, X, o, b
00818     if ((*format != 'i') && (*format != 'd'))
00819     {
00820         flags &= ~(FLAGS_PLUS | FLAGS_SPACE);
00821     }
00822
00823     // ignore '0' flag when precision is given
00824     if (flags & FLAGS_PRECISION)
00825     {
00826         flags &= ~FLAGS_ZEROPAD;
00827     }
00828
00829     // convert the integer
00830     if ((*format == 'i') || (*format == 'd'))
00831     {
00832         // signed
00833         if (flags & FLAGS_LONG_LONG)
00834         {
00835 #if defined(PRINTF_SUPPORT_LONG_LONG)
00836             const long long value = va_arg(va, long long);
00837             idx = _ntoa_long_long(out, buffer, idx, maxlen, (unsigned long long)(value > 0 ?
00838             value : 0 - value), value < 0, base, precision, width, flags);
00839         }
00840         else if (flags & FLAGS_LONG)
00841         {
00842             const long value = va_arg(va, long);
00843             idx = _ntoa_long(out, buffer, idx, maxlen, (unsigned long)(value > 0 ? value : 0 -
00844             value), value < 0, base, precision, width, flags);
00845         }
00846     }

```

```

00847         const int value = (flags & FLAGS_CHAR) ? (char)va_arg(va, int) : (flags &
00848         FLAGS_SHORT) ? (short int)va_arg(va, int)
00849         : va_arg(va, int);
00850         idx = _ntoa_long(out, buffer, idx, maxlen, (unsigned int)(value > 0 ? value : 0 -
00851             value), value < 0, base, precision, width, flags);
00852     }
00853     else
00854     {
00855         // unsigned
00856         if (flags & FLAGS_LONG_LONG)
00857         {
00858 #if defined(PRINTF_SUPPORT_LONG_LONG)
00859             idx = _ntoa_long_long(out, buffer, idx, maxlen, va_arg(va, unsigned long long),
00860             false, base, precision, width, flags);
00861         }
00862         else if (flags & FLAGS_LONG)
00863         {
00864             idx = _ntoa_long(out, buffer, idx, maxlen, va_arg(va, unsigned long), false, base,
00865             precision, width, flags);
00866         }
00867         else
00868         {
00869             const unsigned int value = (flags & FLAGS_CHAR) ? (unsigned char)va_arg(va,
00870             unsigned int) : (flags & FLAGS_SHORT) ? (unsigned short int)va_arg(va, unsigned int)
00871             : va_arg(va, unsigned int);
00872             idx = _ntoa_long(out, buffer, idx, maxlen, value, false, base, precision, width,
00873             flags);
00874         }
00875 #if defined(PRINTF_SUPPORT_FLOAT)
00876         case 'f':
00877         case 'F':
00878             if (*format == 'F')
00879                 flags |= FLAGS_UPPERCASE;
00880             idx = _ftoa(out, buffer, idx, maxlen, va_arg(va, double), precision, width, flags);
00881             format++;
00882             break;
00883 #if defined(PRINTF_SUPPORT_EXPONENTIAL)
00884         case 'e':
00885         case 'E':
00886         case 'g':
00887         case 'G':
00888             if ((*format == 'g') || (*format == 'G'))
00889                 flags |= FLAGS_ADAPT_EXP;
00890             if ((*format == 'E') || (*format == 'G'))
00891                 flags |= FLAGS_UPPERCASE;
00892             idx = _etoa(out, buffer, idx, maxlen, va_arg(va, double), precision, width, flags);
00893             format++;
00894             break;
00895 #endif // PRINTF_SUPPORT_EXPONENTIAL
00896 #endif // PRINTF_SUPPORT_FLOAT
00897         case 'c':
00898         {
00899             unsigned int l = 1U;
00900             // pre padding
00901             if (!(flags & FLAGS_LEFT))
00902             {
00903                 while (l++ < width)
00904                 {
00905                     out(' ', buffer, idx++, maxlen);
00906                 }
00907             }
00908             // char output
00909             out((char)va_arg(va, int), buffer, idx++, maxlen);
00910             // post padding
00911             if (flags & FLAGS_LEFT)
00912             {
00913                 while (l++ < width)
00914                 {
00915                     out(' ', buffer, idx++, maxlen);
00916                 }
00917             }
00918             format++;
00919             break;
00920         }
00921         case 's':
00922         {
00923             const char *p = va_arg(va, char *);
00924             unsigned int l = _strnlen_s(p, precision ? precision : (size_t)-1);

```

```

00926         // pre padding
00927         if (flags & FLAGS_PRECISION)
00928         {
00929             l = (l < precision ? l : precision);
00930         }
00931         if (!(flags & FLAGS_LEFT))
00932         {
00933             while (l++ < width)
00934             {
00935                 out(' ', buffer, idx++, maxlen);
00936             }
00937         }
00938         // string output
00939         while ((*p != 0) && !(flags & FLAGS_PRECISION) || precision--)
00940         {
00941             out(*(p++), buffer, idx++, maxlen);
00942         }
00943         // post padding
00944         if (flags & FLAGS_LEFT)
00945         {
00946             while (l++ < width)
00947             {
00948                 out(' ', buffer, idx++, maxlen);
00949             }
00950         }
00951         format++;
00952         break;
00953     }
00954
00955     case 'p':
00956     {
00957         width = sizeof(void *) * 2U;
00958         flags |= FLAGS_ZEROPAD | FLAGS_UPPERCASE;
00959 #if defined(PRINTF_SUPPORT_LONG_LONG)
00960         const bool is_ll = sizeof(uintptr_t) == sizeof(long long);
00961         if (is_ll)
00962         {
00963             idx = _ntoa_long_long(out, buffer, idx, maxlen, (uintptr_t)va_arg(va, void *), false,
00964             16U, precision, width, flags);
00965         }
00966         else
00967         {
00968             idx = _ntoa_long(out, buffer, idx, maxlen, (unsigned long)((uintptr_t)va_arg(va, void *
00969             *)), false, 16U, precision, width, flags);
00970         }
00971 #endif
00972         format++;
00973         break;
00974     }
00975
00976     case '%':
00977     {
00978         out('%', buffer, idx++, maxlen);
00979         format++;
00980         break;
00981     default:
00982     {
00983         out(*format, buffer, idx++, maxlen);
00984         format++;
00985         break;
00986     }
00987
00988     // termination
00989     out((char)0, buffer, idx < maxlen ? idx : maxlen - 1U, maxlen);
00990
00991     // return written chars without terminating \0
00992     return (int)idx;
00993 }
00994
00995
00996
00997 int printf_(const char *format, ...)
00998 {
00999     va_list va;
01000     va_start(va, format);
01001     char buffer[1];
01002     const int ret = _vsnprintf(_out_char, buffer, (size_t)-1, format, va);
01003     va_end(va);
01004     return ret;
01005 }
01006
01007 int sprintf_(char *buffer, const char *format, ...)
01008 {
01009     va_list va;
01010     va_start(va, format);
01011     const int ret = _vsnprintf(_out_buffer, buffer, (size_t)-1, format, va);

```

```

01012     va_end(va);
01013     return ret;
01014 }
01015
01016 int sprintf_(char *buffer, size_t count, const char *format, ...)
01017 {
01018     va_list va;
01019     va_start(va, format);
01020     const int ret = _vsnprintf(_out_buffer, buffer, count, format, va);
01021     va_end(va);
01022     return ret;
01023 }
01024
01025 int vprintf_(const char *format, va_list va)
01026 {
01027     char buffer[1];
01028     return _vsnprintf(_out_char, buffer, (size_t)-1, format, va);
01029 }
01030
01031 int vsnprintf_(char *buffer, size_t count, const char *format, va_list va)
01032 {
01033     return _vsnprintf(_out_buffer, buffer, count, format, va);
01034 }
01035
01036 int fctprintf(void (*out)(char character, void *arg), void *arg, const char *format, ...)
01037 {
01038     va_list va;
01039     va_start(va, format);
01040     const out_fct_wrap_type out_fct_wrap = {out, arg};
01041     const int ret = _vsnprintf(_out_fct, (char *) (uintptr_t) &out_fct_wrap, (size_t)-1, format, va);
01042     va_end(va);
01043     return ret;
01044 }

```

## 4.119 src/proc.c File Reference

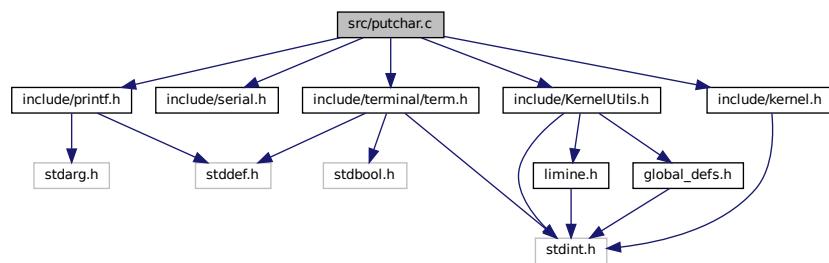
## 4.120 proc.c

## 4.121 src/putchar.c File Reference

```

#include "include/printf.h"
#include "include/serial.h"
#include "include/kernel.h"
#include "include/terminal/term.h"
#include "include/KernelUtils.h"
Include dependency graph for putchar.c:

```



## Functions

- void \_putchar (char character)

## 4.121.1 Function Documentation

### 4.121.1.1 `_putchar()`

```
void _putchar (
    char character )
```

Output a character to a custom device like UART, used by the `printf()` function. This function is declared here only. You have to write your custom implementation somewhere.

#### Parameters

<code>character</code>	Character to output
------------------------	---------------------

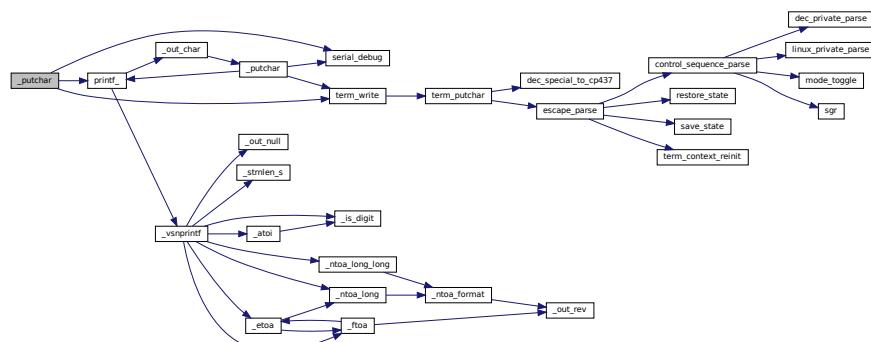
lets the printf function use the serial or terminal depending if the kernel has setup memory for the terminal or not.

#### Parameters

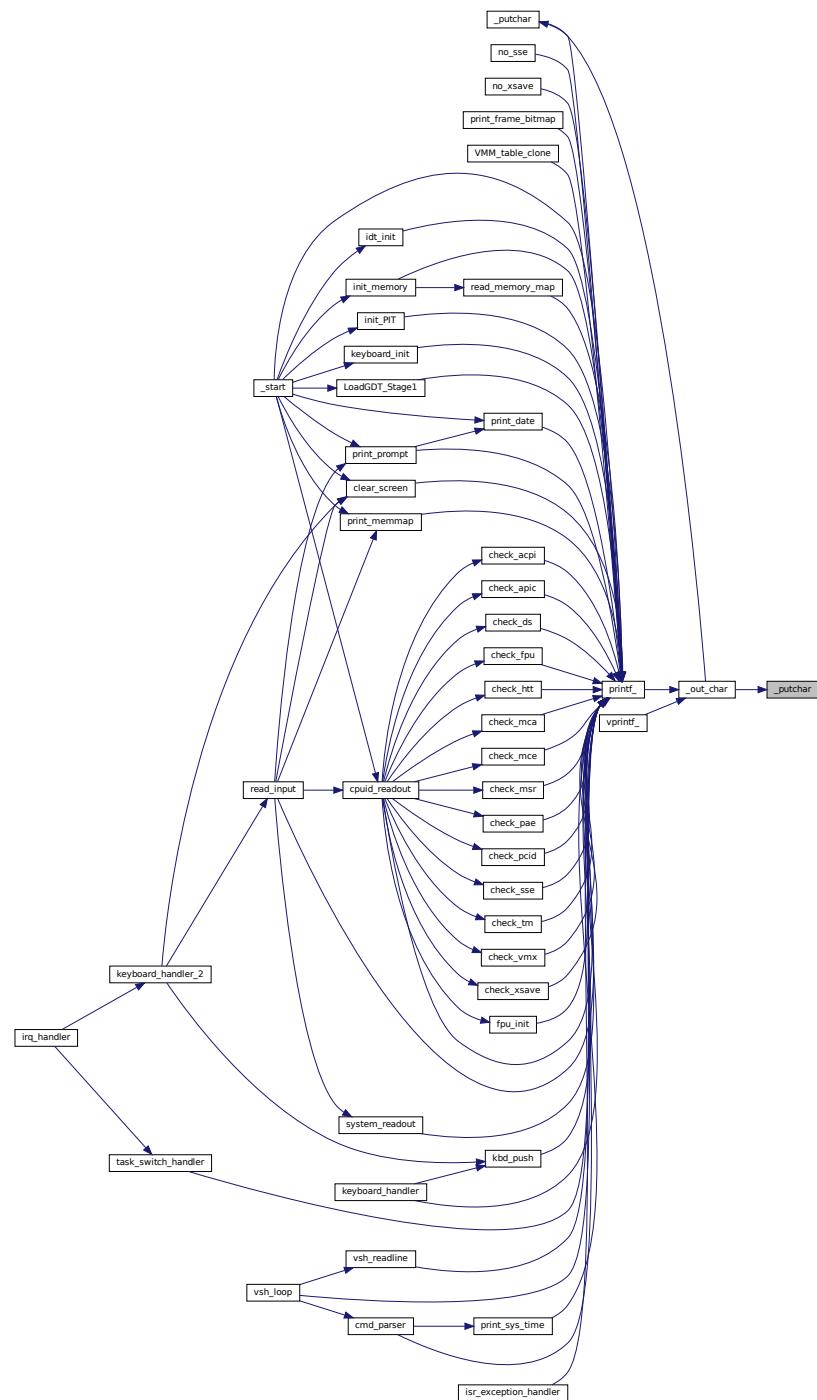
<code>character</code>	
------------------------	--

Definition at line 7 of file `putchar.c`.

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.122 putchar.c

```

00001 #include "include/printf.h"
00002 #include "include/serial.h"
00003 #include "include/kernel.h"
00004 #include "include/terminal/term.h"
00005 #include "include/KernelUtils.h"
00006
00007 void _putchar(char character)

```

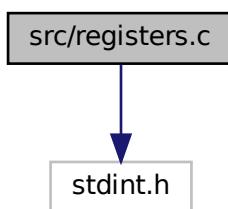
```

00008 {
00009
00013     if (kerror_mode == 1)
00014     {
00015
00016         term_write(term_context, &character, sizeof(char));
00017         serial_debug(character);
00018     }
00019     else if (kerror_mode == 2)
00020     {
00021
00022         serial_debug(character);
00023     }
00024     else if (bootspace == 1)
00025     {
00026
00027         serial_debug(character);
00028     }
00029     else if (bootspace == 2)
00030     {
00031
00032         early_term.response->write(early_term.response->terminals[0], &character, sizeof(char));
00033         serial_debug(character);
00034     }
00035
00036     else if (bootspace == 3)
00037     {
00038
00039         early_term.response->write(early_term.response->terminals[0], &character, sizeof(char));
00040     }
00041
00042     else
00043     {
00044         if (term_context)
00045         {
00046
00047             term_write(term_context, &character, sizeof(char));
00048         }
00049         else
00050         {
00051
00052             serial_debug(character);
00053             printf_("%s\n", "ERROR: TERMINAL WRITE FAILURE!");
00054             printf_("%s\n", "Defaulting to serial");
00055
00056             return;
00057         }
00058     }
00059
00060 // serial_debug(character);
00061 }

```

## 4.123 src/registers.c File Reference

#include <stdint.h>  
Include dependency graph for registers.c:



## Functions

- void [writeMSR](#) (uint64\_t msr, uint64\_t value)

### 4.123.1 Function Documentation

#### 4.123.1.1 writeMSR()

```
void writeMSR (
    uint64_t msr,
    uint64_t value )
```

Definition at line 3 of file [registers.c](#).

Here is the caller graph for this function:



## 4.124 registers.c

```
00001 #include <stdint.h>
00002
00003 void writeMSR(uint64_t msr, uint64_t value)
00004 {
00005     uint32_t lo = (uint32_t)value;
00006     uint32_t hi = value >> 32;
00007
00008     asm volatile("wrmsr"
00009                 : /* no output */
00010                 : "a"(lo), "d"(hi), "c"(msr));
00011 }
```

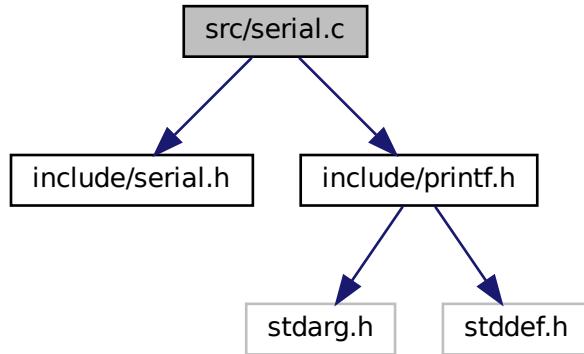
## 4.125 src/sched.c File Reference

## 4.126 sched.c

## 4.127 src/serial.c File Reference

```
#include "include/serial.h"
#include "include/printf.h"
```

Include dependency graph for serial.c:



## Functions

- void [serial\\_print](#) (const char \*str)
- void [serial\\_print\\_line](#) (const char \*str)

### 4.127.1 Function Documentation

#### 4.127.1.1 [serial\\_print\(\)](#)

```
void serial_print (
    const char * str )
```

Definition at line 4 of file [serial.c](#).

Here is the call graph for this function:

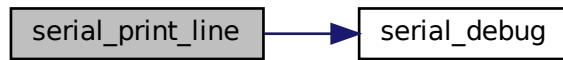


### 4.127.1.2 serial\_print\_line()

```
void serial_print_line (
    const char * str )
```

Definition at line 14 of file [serial.c](#).

Here is the call graph for this function:



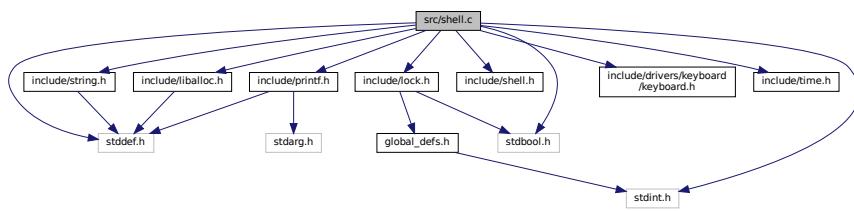
## 4.128 serial.c

```
00001 #include "include/serial.h"
00002 #include "include/printf.h"
00003
00004 void serial_print(const char *str)
00005 {
00006     char *p = (char *)str;
00007
00008     while (*p)
00009     {
00010         serial_debug(*p++);
00011     }
00012 }
00013
00014 void serial_print_line(const char *str)
00015 {
00016     char *p = (char *)str;
00017
00018     while (*p)
00019     {
00020         serial_debug(*p++);
00021     }
00022
00023     serial_debug('\n');
00024 }
```

## 4.129 src/shell.c File Reference

```
#include <stddef.h>
#include <stdint.h>
#include <stdbool.h>
#include "include/string.h"
#include "include/liballoc.h"
#include "include/shell.h"
#include "include/printf.h"
#include "include/drivers/keyboard/keyboard.h"
#include "include/time.h"
```

```
#include "include/lock.h"
Include dependency graph for shell.c:
```



## Functions

- void [vsh\\_loop \(\)](#)
- char [vsh\\_readline \(\)](#)
- void [cmd\\_parser \(const char str\)](#)

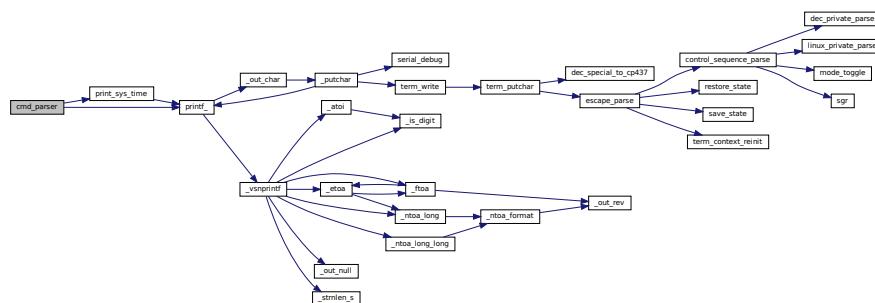
### 4.129.1 Function Documentation

#### 4.129.1.1 cmd\_parser()

```
void cmd_parser (
    const char str )
```

Definition at line 85 of file [shell.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

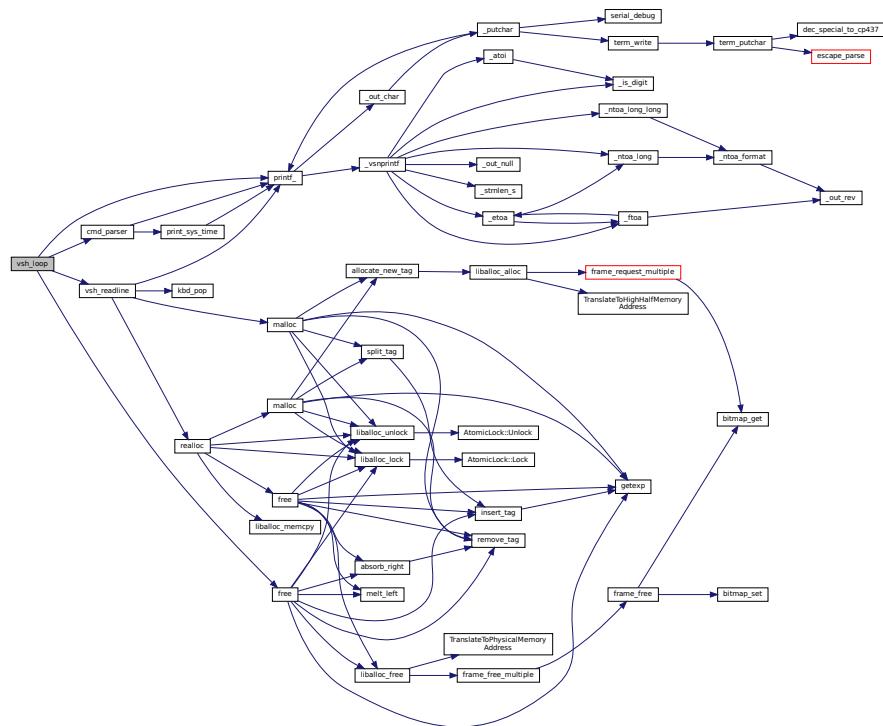


### 4.129.1.2 vsh\_loop()

```
void vsh_loop ( )
```

Definition at line 12 of file [shell.c](#).

Here is the call graph for this function:

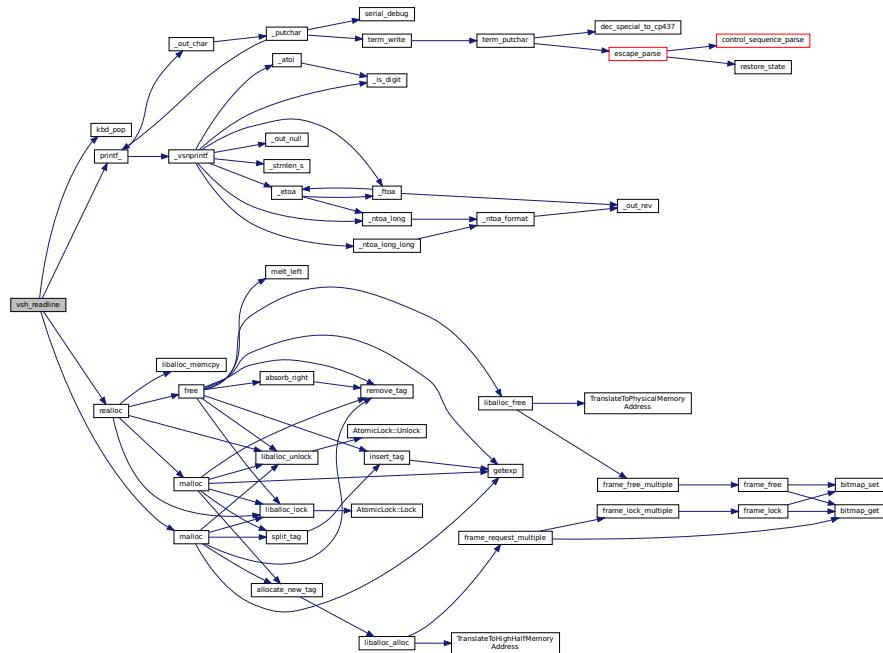


### 4.129.1.3 vsh\_readline()

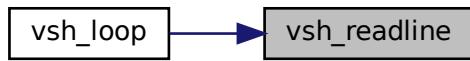
```
char vsh_readline ( )
```

Definition at line 28 of file [shell.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.130 shell.c

```

00001 #include <stddef.h>
00002 #include <stdint.h>
00003 #include <stdbool.h>
00004 #include "include/string.h"
00005 #include "include/liballoc.h"
00006 #include "include/shell.h"
00007 #include "include/printf.h"
00008 #include "include/drivers/keyboard/keyboard.h"
00009 #include "include/time.h"
00010 #include "include/lock.h"
00011
00012 void vsh_loop()
00013 {
00014     char *line;
00015     int status;
00016
00017     printf_(""%s", ":> ");
00018
00019     line = vsh_readline();
00020
00021     cmd_parser(line);

```

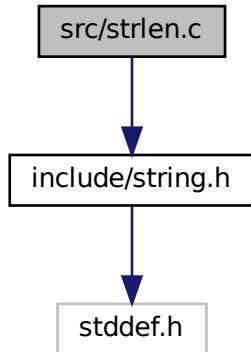
```

00024     free(line);
00025 }
00026 }
00027
00028 char vsh_readline()
00029 {
00030
00031     int bufsize = VSH_CMD_BUFFER_SIZE;
00032     int pos = 0;
00033     char *buffer = malloc(sizeof(char) * bufsize);
00034     int c;
00035
00036     if (!buffer)
00037     {
00038
00039         printf_("\"%s\\n\", \"vsh: Command Buffer Allocation Error!\"");
00040         // return;
00041     }
00042
00043     while (1)
00044     {
00045
00046     L1:
00047
00048         c = kbd_pop();
00049
00050         if (c == 0)
00051         {
00052
00053             goto L1;
00054         }
00055
00056         if (c == '+')
00057         {
00058
00059             return buffer;
00060         }
00061         else
00062         {
00063
00064             buffer[pos] = c;
00065         }
00066
00067         pos++;
00068
00069         if (pos >= bufsize)
00070         {
00071
00072             bufsize += VSH_CMD_BUFFER_SIZE;
00073             buffer = realloc(buffer, bufsize);
00074
00075             if (!buffer)
00076             {
00077
00078                 printf_("\"%s\\n\", \"vsh: Command Buffer Allocation Error!\"");
00079                 // return;
00080             }
00081         }
00082     }
00083 }
00084
00085 void cmd_parser(const char str)
00086 {
00087
00088     if (str == "time")
00089     {
00090
00091         print_sys_time();
00092     }
00093     else
00094     {
00095
00096         printf_("\"%s\\n\", \"unknown command\"");
00097     }
00098 }
```

## 4.131 src/strlen.c File Reference

```
#include "include/string.h"
```

Include dependency graph for `strlen.c`:



## Functions

- `size_t strlen (const char *str)`

### 4.131.1 Function Documentation

#### 4.131.1.1 [strlen\(\)](#)

```
size_t strlen (
    const char * str )
```

Definition at line 3 of file [strlen.c](#).

Here is the caller graph for this function:

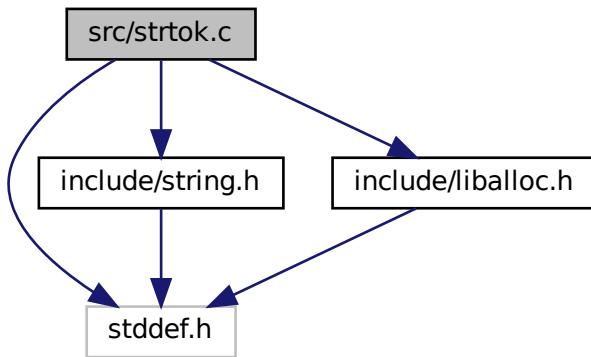


## 4.132 strlen.c

```
00001 #include "include/string.h"
00002
00003 size_t strlen(const char *str)
00004 {
00005     size_t len = 0;
00006     while (str[len])
00007         len++;
00008     return len;
00009 }
```

## 4.133 src/strtok.c File Reference

```
#include <stddef.h>
#include "include/string.h"
#include "include/liballoc.h"
Include dependency graph for strtok.c:
```



### Functions

- int **strtok** (char \*srcstr, char sep, char \*\*\*output)

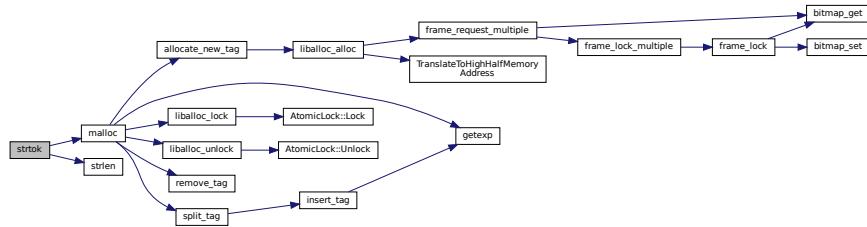
#### 4.133.1 Function Documentation

##### 4.133.1.1 strtok()

```
int strtok (
    char * srcstr,
    char sep,
    char *** output )
```

Definition at line 5 of file `strtok.c`.

Here is the call graph for this function:



## 4.134 strtok.c

```

00001 #include <stddef.h>
00002 #include "include/string.h"
00003 #include "include/liballoc.h"
00004
00005 int strtok(char *srcstr, char sep, char ***output)
00006 {
00007     /* Assume we have a properly NULL-terminated string. */
00008     int len = strlen(srcstr);
00009
00010     /* Initialize the number of tokens to 0. */
00011     int numparts = 0;
00012
00013     /* A temporary pointer to a char pointer we'll use later. */
00014     char **currentpart;
00015
00016     /*
00017     * Replace every instance of sep with a NULL character.
00018     * This has the effect of turning one string into many; you can
00019     * think of the first string beginning where the original string
00020     * did, and ending at the first NULL, then the second string
00021     * beginning at the character right after that NULL.
00022     */
00023     int i;
00024     for (i = 0; i < len; i++)
00025     {
00026         if (srcstr[i] == sep)
00027         {
00028             srcstr[i] = '\0';
00029             numparts++;
00030         }
00031     }
00032
00033     /*
00034     * Increment numparts one more time, for the last substring.
00035     * i.e. if we found N separators we have N+1 parts. If two
00036     * separator characters appear together, or there is a
00037     * separator at the end, then some of these become the empty
00038     * string "".
00039     */
00040     numparts++;
00041
00042     /* Allocate space for our pointers to the various substrings
00043     * Note that here we dereference output once, to assign to the
00044     * char ** we declared in the calling function (i.e. main())
00045     */
00046     *output = malloc(numparts * sizeof(char *));
00047
00048     /* You should always check the return value of malloc! */
00049     /* if (*output == NULL)
00050     {
00051         perror("malloc() call in str_separate");
00052         exit(1);
00053     } */
00054
00055     /* Start with currentpart pointing to the first allocated pointer.
00056     * We'll use this to step through the array of char * pointers.
00057     */
00058     currentpart = *output;
00059

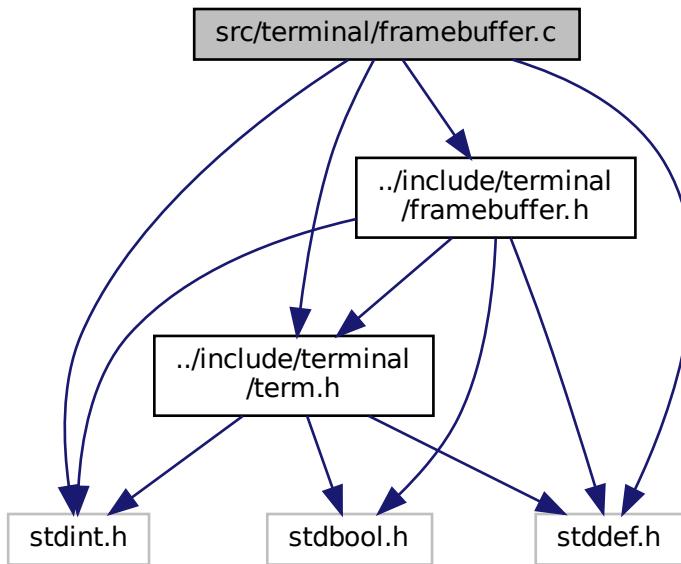
```

```
00060      /*
00061      * The first char * pointer should point to the beginning of the
00062      * original string, as that is necessarily the first token.
00063      */
00064  *currentpart = srcstr;
00065
00066  /*
00067  * We've placed \0's in the place of every occurrence of sep; so
00068  * a new token/substring begins at the character AFTER a \0
00069  * (excluding the very last \0, which we will never hit with this
00070  * loop).
00071  *
00072  * This loop finds all the NULLs we placed in the string, and stores
00073  * a pointer to the character after each NULL in *currentpart, after
00074  * incrementing currentpart to move to the next char * slot.
00075  */
00076
00077  for (i = 0; i < len; i++) /* < len: we never get to the last '\0',
00078  * since strlen gives us the length of
00079  * the string, not including the NULL */
00080  {
00081      if (srcstr[i] == '\0')
00082      {
00083          currentpart++;
00084          *currentpart = &(srcstr[i + 1]);
00085      }
00086  }
00087
00088  /*
00089  * We only return the integer numparts, but in fact we are
00090  * "implicitly returning" the contents of *output, since this
00091  * was declared outside str_separate but modified by it.
00092  */
00093  return numparts;
00094 }
```

## 4.135 src/terminal/framebuffer.c File Reference

```
#include <stdint.h>
#include <stddef.h>
#include "../include/terminal/term.h"
#include "../include/terminal/framebuffer.h"
```

Include dependency graph for framebuffer.c:



## Macros

- `#define FONT_BYTES ((font_width * font_height * FBTERM_FONT_GLYPHS) / 8)`

## Functions

- `void * memset (void *, int, size_t)`
- `void * memcpy (void *, const void *, size_t)`
- `static void fbterm_save_state (struct term_context *_ctx)`
- `static void fbterm_restore_state (struct term_context *_ctx)`
- `static void fbterm_swap_palette (struct term_context *_ctx)`
- `static void plot_char (struct term_context *_ctx, struct fbterm_char *c, size_t x, size_t y)`
- `static void plot_char_fast (struct term_context *_ctx, struct fbterm_char *old, struct fbterm_char *c, size_t x, size_t y)`
- `static bool compare_char (struct fbterm_char *a, struct fbterm_char *b)`
- `static void push_to_queue (struct term_context *_ctx, struct fbterm_char *c, size_t x, size_t y)`
- `static void fbterm_revscroll (struct term_context *_ctx)`
- `static void fbterm_scroll (struct term_context *_ctx)`
- `static void fbterm_clear (struct term_context *_ctx, bool move)`
- `static void fbterm_enable_cursor (struct term_context *_ctx)`
- `static bool fbterm_disable_cursor (struct term_context *_ctx)`
- `static void fbterm_set_cursor_pos (struct term_context *_ctx, size_t x, size_t y)`
- `static void fbterm_get_cursor_pos (struct term_context *_ctx, size_t *x, size_t *y)`
- `static void fbterm_move_character (struct term_context *_ctx, size_t new_x, size_t new_y, size_t old_x, size_t old_y)`
- `static void fbterm_set_text_fg (struct term_context *_ctx, size_t fg)`

- static void `fbterm_set_text_bg` (struct `term_context` \*\_ctx, size\_t bg)
- static void `fbterm_set_text_fg_bright` (struct `term_context` \*\_ctx, size\_t fg)
- static void `fbterm_set_text_bg_bright` (struct `term_context` \*\_ctx, size\_t bg)
- static void `fbterm_set_text_fg_rgb` (struct `term_context` \*\_ctx, uint32\_t fg)
- static void `fbterm_set_text_bg_rgb` (struct `term_context` \*\_ctx, uint32\_t bg)
- static void `fbterm_set_text_fg_default` (struct `term_context` \*\_ctx)
- static void `fbterm_set_text_bg_default` (struct `term_context` \*\_ctx)
- static void `draw_cursor` (struct `term_context` \*\_ctx)
- static void `fbterm_double_buffer_flush` (struct `term_context` \*\_ctx)
- static void `fbterm_raw_putchar` (struct `term_context` \*\_ctx, uint8\_t c)
- static void `fbterm_full_refresh` (struct `term_context` \*\_ctx)
- static void `fbterm_deinit` (struct `term_context` \*\_ctx, void(\*\_free)(void \*, size\_t))
- struct `term_context` \* `fbterm_init` (void \*(\*\_malloc)(size\_t), uint32\_t \*framebuffer, size\_t width, size\_t height, size\_t pitch, uint32\_t \*canvas, uint32\_t \*ansi\_colours, uint32\_t \*ansi\_bright\_colours, uint32\_t \*default\_bg, uint32\_t \*default\_fg, void \*font, size\_t font\_width, size\_t font\_height, size\_t font\_spacing, size\_t font\_scale\_x, size\_t font\_scale\_y, size\_t margin)

## Variables

- static const uint8\_t `builtin_font` []

### 4.135.1 Macro Definition Documentation

#### 4.135.1.1 FONT\_BYTES

```
#define FONT_BYTES ((font_width * font_height * FBTERM_FONT_GLYPHS) / 8)
```

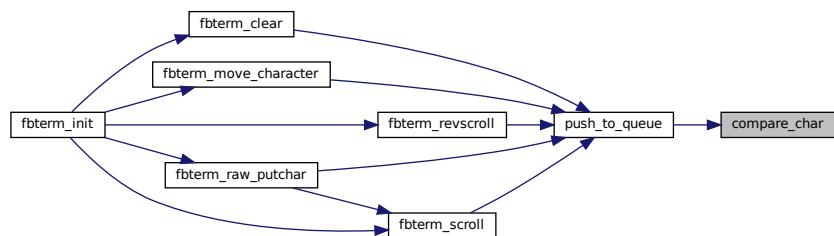
### 4.135.2 Function Documentation

#### 4.135.2.1 compare\_char()

```
static bool compare_char (
    struct fbterm_char * a,
    struct fbterm_char * b ) [inline], [static]
```

Definition at line 353 of file [framebuffer.c](#).

Here is the caller graph for this function:



#### 4.135.2.2 draw\_cursor()

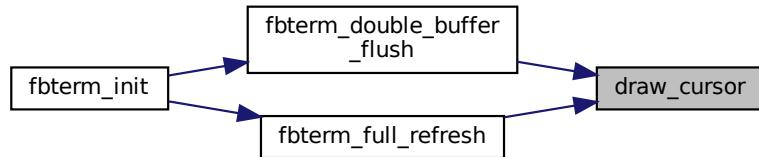
```
static void draw_cursor (
    struct term_context * _ctx ) [static]
```

Definition at line 563 of file [framebuffer.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

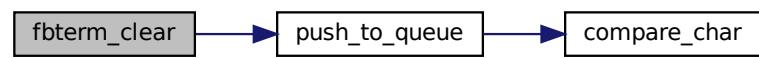


#### 4.135.2.3 fbterm\_clear()

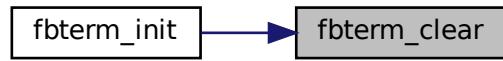
```
static void fbterm_clear (
    struct term_context * _ctx,
    bool move ) [static]
```

Definition at line 435 of file [framebuffer.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.135.2.4 fbterm\_deinit()

```
static void fbterm_deinit (
    struct term_context * _ctx,
    void(*)(void *, size_t) _free ) [static]
```

Definition at line 656 of file [framebuffer.c](#).

Here is the caller graph for this function:

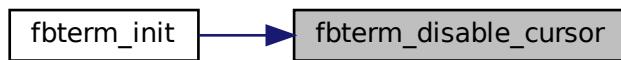


#### 4.135.2.5 fbterm\_disable\_cursor()

```
static bool fbterm_disable_cursor (
    struct term_context * _ctx ) [static]
```

Definition at line 458 of file [framebuffer.c](#).

Here is the caller graph for this function:

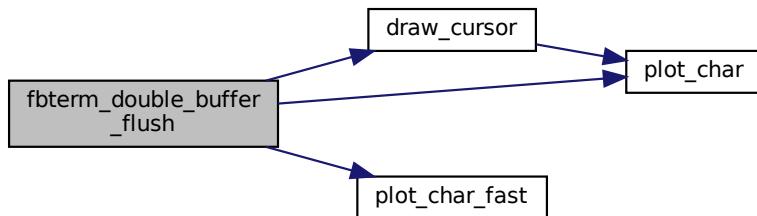


#### 4.135.2.6 fbterm\_double\_buffer\_flush()

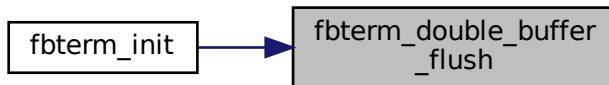
```
static void fbterm_double_buffer_flush (
    struct term_context * _ctx ) [static]
```

Definition at line 584 of file [framebuffer.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.135.2.7 fbterm\_enable\_cursor()

```
static void fbterm_enable_cursor (
    struct term_context * _ctx ) [static]
```

Definition at line 452 of file [framebuffer.c](#).

Here is the caller graph for this function:

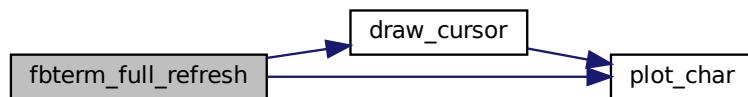


#### 4.135.2.8 fbterm\_full\_refresh()

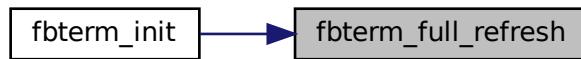
```
static void fbterm_full_refresh (
    struct term_context * _ctx ) [static]
```

Definition at line 635 of file [framebuffer.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.135.2.9 fbterm\_get\_cursor\_pos()

```
static void fbterm_get_cursor_pos (
    struct term_context * _ctx,
    size_t * x,
    size_t * y ) [static]
```

Definition at line 487 of file [framebuffer.c](#).

Here is the caller graph for this function:

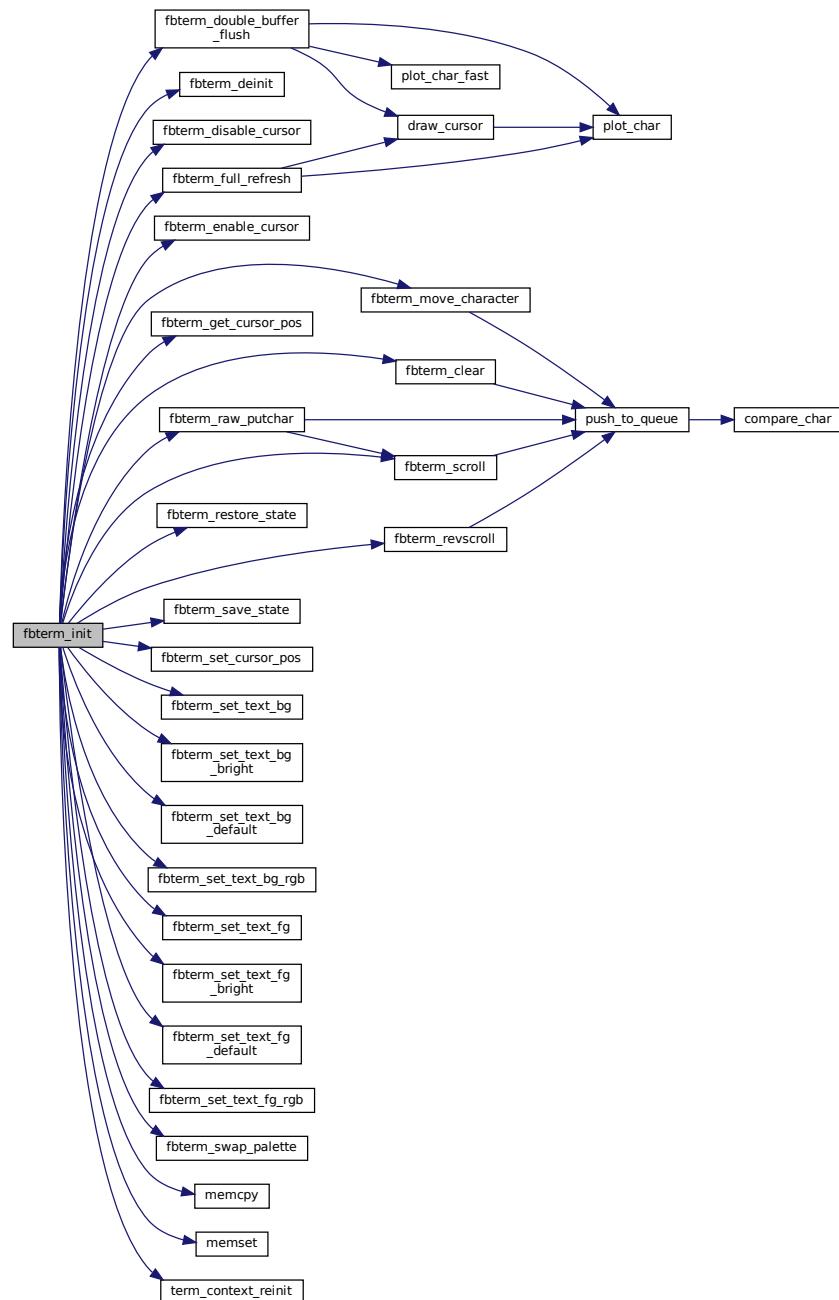


#### 4.135.2.10 fbterm\_init()

```
struct term_context* fbterm_init (
    void *(*)(size_t) _malloc,
    uint32_t * framebuffer,
    size_t width,
    size_t height,
    size_t pitch,
    uint32_t * canvas,
    uint32_t * ansi_colours,
    uint32_t * ansi_bright_colours,
    uint32_t * default_bg,
    uint32_t * default_fg,
    void * font,
    size_t font_width,
    size_t font_height,
    size_t font_spacing,
    size_t font_scale_x,
    size_t font_scale_y,
    size_t margin )
```

Definition at line 668 of file [framebuffer.c](#).

Here is the call graph for this function:



#### 4.135.2.11 `fbterm_move_character()`

```

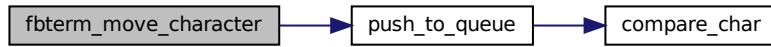
static void fbterm_move_character (
    struct term_context * _ctx,
    size_t new_x,
    size_t new_y,

```

```
size_t old_x,
size_t old_y ) [static]
```

Definition at line 494 of file [framebuffer.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

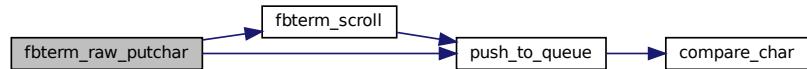


#### 4.135.2.12 fbterm\_raw\_putchar()

```
static void fbterm_raw_putchar (
    struct term_context * _ctx,
    uint8_t c ) [static]
```

Definition at line 617 of file [framebuffer.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.135.2.13 fbterm\_restore\_state()

```
static void fbterm_restore_state (
    struct term_context * _ctx ) [static]
```

Definition at line 279 of file [framebuffer.c](#).

Here is the caller graph for this function:

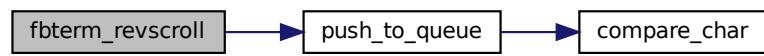


#### 4.135.2.14 fbterm\_revscroll()

```
static void fbterm_revscroll (
    struct term_context * _ctx ) [static]
```

Definition at line 381 of file [framebuffer.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.135.2.15 fbterm\_save\_state()

```
static void fbterm_save_state (
    struct term_context * _ctx ) [static]
```

Definition at line 271 of file [framebuffer.c](#).

Here is the caller graph for this function:

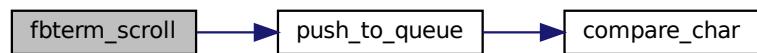


#### 4.135.2.16 fbterm\_scroll()

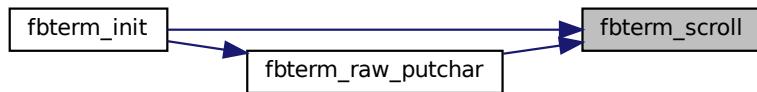
```
static void fbterm_scroll (
    struct term_context * _ctx ) [static]
```

Definition at line 409 of file [framebuffer.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.135.2.17 fbterm\_set\_cursor\_pos()

```
static void fbterm_set_cursor_pos (
    struct term_context * _ctx,
    size_t x,
    size_t y ) [static]
```

Definition at line 466 of file [framebuffer.c](#).

Here is the caller graph for this function:



#### 4.135.2.18 fbterm\_set\_text\_bg()

```
static void fbterm_set_text_bg (
    struct term_context * _ctx,
    size_t bg ) [static]
```

Definition at line 521 of file [framebuffer.c](#).

Here is the caller graph for this function:



#### 4.135.2.19 fbterm\_set\_text\_bg\_bright()

```
static void fbterm_set_text_bg_bright (
    struct term_context * _ctx,
    size_t bg ) [static]
```

Definition at line 533 of file [framebuffer.c](#).

Here is the caller graph for this function:



#### 4.135.2.20 fbterm\_set\_text\_bg\_default()

```
static void fbterm_set_text_bg_default (
    struct term_context * _ctx ) [static]
```

Definition at line 557 of file [framebuffer.c](#).

Here is the caller graph for this function:



**4.135.2.21 fbterm\_set\_text\_bg\_rgb()**

```
static void fbterm_set_text_bg_rgb (
    struct term_context * _ctx,
    uint32_t bg ) [static]
```

Definition at line 545 of file [framebuffer.c](#).

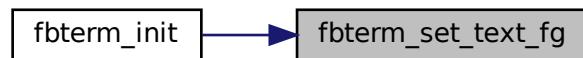
Here is the caller graph for this function:

**4.135.2.22 fbterm\_set\_text\_fg()**

```
static void fbterm_set_text_fg (
    struct term_context * _ctx,
    size_t fg ) [static]
```

Definition at line 515 of file [framebuffer.c](#).

Here is the caller graph for this function:

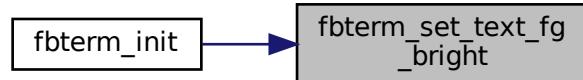


#### 4.135.2.23 fbterm\_set\_text\_fg\_bright()

```
static void fbterm_set_text_fg_bright (
    struct term_context * _ctx,
    size_t fg ) [static]
```

Definition at line 527 of file [framebuffer.c](#).

Here is the caller graph for this function:

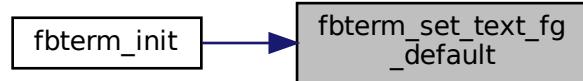


#### 4.135.2.24 fbterm\_set\_text\_fg\_default()

```
static void fbterm_set_text_fg_default (
    struct term_context * _ctx ) [static]
```

Definition at line 551 of file [framebuffer.c](#).

Here is the caller graph for this function:



#### 4.135.2.25 fbterm\_set\_text\_fg\_rgb()

```
static void fbterm_set_text_fg_rgb (
    struct term_context * _ctx,
    uint32_t fg ) [static]
```

Definition at line 539 of file [framebuffer.c](#).

Here is the caller graph for this function:



#### 4.135.2.26 fbterm\_swap\_palette()

```
static void fbterm_swap_palette (
    struct term_context * _ctx ) [static]
```

Definition at line 287 of file [framebuffer.c](#).

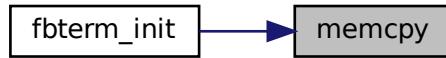
Here is the caller graph for this function:



#### 4.135.2.27 `memcpy()`

```
void* memcpy (
    void * ,
    const void * ,
    size_t )
```

Here is the caller graph for this function:



#### 4.135.2.28 `memset()`

```
void* memset (
    void * bufptr,
    int value,
    size_t size )
```

Definition at line 3 of file [memset.c](#).

Here is the caller graph for this function:

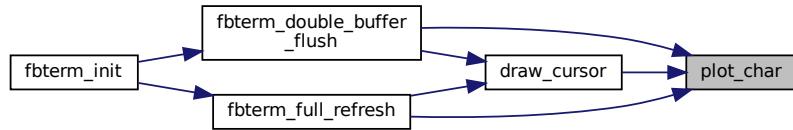


#### 4.135.2.29 plot\_char()

```
static void plot_char (
    struct term_context * _ctx,
    struct fbterm_char * c,
    size_t x,
    size_t y ) [static]
```

Definition at line 294 of file [framebuffer.c](#).

Here is the caller graph for this function:

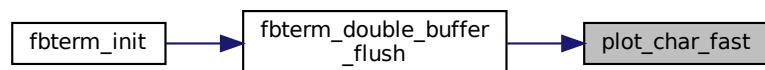


#### 4.135.2.30 plot\_char\_fast()

```
static void plot_char_fast (
    struct term_context * _ctx,
    struct fbterm_char * old,
    struct fbterm_char * c,
    size_t x,
    size_t y ) [static]
```

Definition at line 322 of file [framebuffer.c](#).

Here is the caller graph for this function:



#### 4.135.2.31 push\_to\_queue()

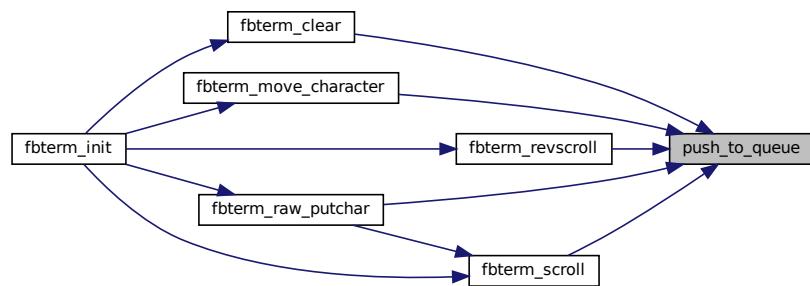
```
static void push_to_queue (
    struct term_context * _ctx,
    struct fbterm_char * c,
    size_t x,
    size_t y ) [static]
```

Definition at line 357 of file [framebuffer.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 4.135.3 Variable Documentation

#### 4.135.3.1 builtin\_font

```
const uint8_t builtin_font[] [static]
```

Definition at line 13 of file [framebuffer.c](#).

## 4.136 framebuffer.c

```

00001 #include <stdint.h>
00002 #include <stddef.h>
00003
00004 #include "../include/terminal/term.h"
00005 #include "../include/terminal/framebuffer.h"
00006
00007 void *memset(void *, int, size_t);
00008 void *memcpy(void *, const void *, size_t);
00009
00010 // Builtin font originally taken from:
00011 // https://github.com/viler-int10h/vga-text-mode-fonts/raw/master/FONTS/PC-OTHER/TOSH-SAT.F16
00012 // array size is 4096
00013 static const uint8_t builtin_font[] = {
00014     0x00, 0x00,
00015     0x00, 0x00, 0x7c, 0x82, 0x82, 0x82, 0xaa, 0x82, 0x82, 0xba, 0x92, 0x82, 0x7c, 0x00, 0x00, 0x00, 0x00,
00016     0x00, 0x00, 0x7c, 0xfe, 0xfe, 0xd6, 0xfe, 0xfe, 0xc6, 0xee, 0xfe, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
00017     0x00, 0x00, 0x00, 0x00, 0x00, 0x6c, 0xfe, 0xfe, 0x7c, 0x38, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00018     0x00, 0x00, 0x00, 0x10, 0x38, 0x7c, 0xfe, 0x7c, 0x38, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00019     0x00, 0x00, 0x00, 0x00, 0x38, 0x38, 0x38, 0xee, 0xee, 0x10, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00,
00020     0x00, 0x00, 0x00, 0x10, 0x38, 0x7c, 0xfe, 0x54, 0x10, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00,
00021     0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x3c, 0x3c, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00022     0xff, 0xff, 0xff, 0xff, 0x7e, 0xc3, 0xc3, 0xe7, 0xff, 0xff, 0xff, 0xff, 0x7e, 0xff, 0x7e, 0x7e,
00023     0x00, 0x00, 0x00, 0x00, 0x3c, 0x66, 0x42, 0x42, 0x66, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00024     0xff, 0xff, 0xff, 0xff, 0xc3, 0x99, 0xbd, 0xbd, 0x99, 0xc3, 0xff, 0xff, 0xff, 0xff, 0x7e, 0x7e,
00025     0x00, 0x00, 0x1e, 0x06, 0xa, 0x10, 0x10, 0x7c, 0x82, 0x82, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
00026     0x00, 0x00, 0x7c, 0x82, 0x82, 0x7c, 0x10, 0x10, 0x7c, 0x10, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00,
00027     0x00, 0x00, 0x1e, 0x1e, 0x1e, 0x10, 0x10, 0x10, 0xf0, 0xf0, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00,
00028     0x00, 0x00, 0x3e, 0x22, 0x3e, 0x22, 0x22, 0x22, 0x2e, 0xee, 0xec, 0xc0, 0x00, 0x00, 0x00, 0x00,
00029     0x00, 0x00, 0x10, 0x92, 0x7c, 0x44, 0x82, 0x82, 0x82, 0x44, 0x7c, 0x92, 0x10, 0x00, 0x00, 0x00,
00030     0x00, 0x00, 0x00, 0x00, 0x00, 0xc0, 0xe0, 0xf0, 0xf0, 0xe0, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00,
00031     0x00, 0x00, 0x00, 0x00, 0x18, 0x38, 0x38, 0x78, 0x78, 0x78, 0x38, 0x18, 0x00, 0x00, 0x00, 0x00,
00032     0x00, 0x00, 0x10, 0x38, 0x54, 0x92, 0x10, 0x10, 0x92, 0x54, 0x38, 0x10, 0x00, 0x00, 0x00, 0x00,
00033     0x00, 0x00, 0x44, 0x44,
00034     0x00, 0x00, 0x7e, 0xf2, 0xf2, 0xf2, 0x72, 0x72, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x00, 0x00,
00035     0x00, 0x78, 0x84, 0x40, 0x30, 0x48, 0x84, 0x84, 0x48, 0x30, 0x08, 0x84, 0x78, 0x00, 0x00, 0x00,
00036     0x00, 0x00,
00037     0x00, 0x00, 0x10, 0x38, 0x54, 0x92, 0x10, 0x10, 0x92, 0x54, 0x38, 0x10, 0x00, 0x00, 0x00, 0x00,
00038     0x00, 0x00, 0x10, 0x38, 0x54, 0x92, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x00, 0x00, 0x00, 0x00,
00039     0x00, 0x00, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x92, 0x54, 0x38, 0x10, 0x00, 0x00, 0x00, 0x00,
00040     0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x04, 0x0fe, 0x04, 0x08, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00,
00041     0x00, 0x00, 0x00, 0x00, 0x10, 0x20, 0x40, 0xfe, 0x40, 0x20, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00,
00042     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00043     0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x42, 0x42, 0x42, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00044     0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x38, 0x7c, 0xfe, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00045     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0x38, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00046     0x00, 0x00,
00047     0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00,
00048     0x00, 0x48, 0x48, 0x48, 0x48, 0x00, 0x00,
00049     0x00, 0x00, 0x44, 0x44,
00050     0x00, 0x10, 0x10, 0x7c, 0x92, 0x90, 0x80, 0x7c, 0x02, 0x12, 0x92, 0x7c, 0x10, 0x10, 0x00, 0x00,
00051     0x00, 0x00, 0x00, 0x60, 0xf2, 0x64, 0x08, 0x10, 0x20, 0x4c, 0x9e, 0xc0, 0x00, 0x00, 0x00, 0x00,
00052     0x00, 0x00, 0x30, 0x48, 0x48, 0x48, 0x32, 0xa, 0x84, 0x84, 0x84, 0x7a, 0x00, 0x00, 0x00, 0x00,
00053     0x00, 0x10, 0x10, 0x20, 0x00, 0x00,
00054     0x00, 0x00, 0x08, 0x10, 0x20, 0x20, 0x20, 0x20, 0x10, 0x08, 0x00, 0x08, 0x10, 0x00, 0x00, 0x00,
00055     0x00, 0x00, 0x20, 0x10, 0x08, 0x08, 0x08, 0x08, 0x08, 0x10, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00,
00056     0x00, 0x00, 0x00, 0x00, 0x10, 0x54, 0x38, 0xfe, 0x38, 0x54, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00,
00057     0x00, 0x00, 0x00, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x00, 0x00, 0x00, 0x00,
00058     0x00, 0x10, 0x10, 0x20, 0x00, 0x00,
00059     0x00, 0x00,
00060     0x00, 0x30, 0x00, 0x00, 0x00, 0x00,
00061     0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x00, 0x00, 0x00, 0x00,
00062     0x00, 0x00, 0x38, 0x44, 0x82, 0x82, 0x92, 0x82, 0x82, 0x44, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00,
00063     0x00, 0x00, 0x10, 0x30, 0x50, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x7c, 0x00, 0x00, 0x00, 0x00,
00064     0x00, 0x00, 0x7c, 0x82, 0x02, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00,
00065     0x00, 0x00, 0x7c, 0x82, 0x02, 0x02, 0x3c, 0x02, 0x02, 0x02, 0x82, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
00066     0x00, 0x00, 0x04, 0x0c, 0x14, 0x24, 0x44, 0x84, 0xfe, 0x04, 0x04, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00,
00067     0x00, 0x00, 0xfe, 0x80, 0x80, 0x80, 0xfc, 0x82, 0x02, 0x02, 0x82, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
00068     0x00, 0x00, 0x7c, 0x80, 0x80, 0x80, 0xfc, 0x82, 0x82, 0x82, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
00069     0x00, 0x00, 0xfe, 0x02, 0x02, 0x04, 0x08, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x00, 0x00, 0x00, 0x00,
00070     0x00, 0x00, 0x7c, 0x82, 0x82, 0x44, 0x7c, 0x82, 0x82, 0x82, 0x82, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
00071     0x00, 0x00, 0x7c, 0x82, 0x82, 0x82, 0x82, 0x7e, 0x02, 0x02, 0x02, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
00072     0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x00, 0x10, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00,
00073     0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x10, 0x00, 0x00, 0x00, 0x10, 0x10, 0x20, 0x00, 0x00, 0x00,
00074     0x00, 0x00, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x08, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00075     0x00, 0x00,
00076     0x00, 0x00, 0x40, 0x20, 0x10, 0x00, 0x08, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x00, 0x00, 0x00, 0x00,
00077     0x00, 0x00, 0x7c, 0x82, 0x82, 0x02, 0x0c, 0x10, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00,
00078     0x00, 0x00, 0x7c, 0x82, 0x82, 0xba, 0xaa, 0x9c, 0x80, 0x80, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
00079     0x00, 0x00, 0x38, 0x44, 0x82, 0x82, 0xfe, 0x82, 0x82, 0x82, 0x82, 0x82, 0x82, 0x00, 0x00, 0x00, 0x00,
00080     0x00, 0x00, 0xfc, 0x82, 0x82, 0xfc, 0x82, 0x82, 0x82, 0x82, 0x82, 0x82, 0x0fc, 0x00, 0x00, 0x00, 0x00,
00081     0x00, 0x00, 0x7c, 0x82, 0x80, 0x00, 0x00, 0x00,
00082     0x00, 0x00, 0xf8, 0x84, 0x82, 0x82, 0x82, 0x82, 0x82, 0x82, 0x84, 0x84, 0x84, 0x84, 0x00, 0x00, 0x00, 0x00,
00083     0x00, 0x00, 0xfe, 0x80, 0x80, 0x80, 0xfc, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00,
00084     0x00, 0x00, 0xfe, 0x80, 0x80, 0x80, 0xfc, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00,
00085     0x00, 0x00, 0x7c, 0x82, 0x82, 0x80, 0x80, 0x9e, 0x82, 0x82, 0x82, 0x82, 0x82, 0x7e, 0x00, 0x00, 0x00, 0x00,
```

```

00086 0x00, 0x00, 0x82, 0x82, 0x82, 0xfe, 0x82, 0x82, 0x82, 0x82, 0x82, 0x00, 0x00, 0x00, 0x00,
00087 0x00, 0x00, 0x7c, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x00, 0x00, 0x00, 0x00,
00088 0x00, 0x00, 0x3e, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x00, 0x00, 0x00, 0x00,
00089 0x00, 0x00, 0x84, 0x88, 0x90, 0xa0, 0xc0, 0xa0, 0x90, 0x88, 0x84, 0x82, 0x00, 0x00, 0x00, 0x00,
00090 0x00, 0x00, 0x80, 0x00, 0x00, 0x00, 0x00,
00091 0x00, 0x00, 0x82, 0xc6, 0xaa, 0x92, 0x92, 0x82, 0x82, 0x82, 0x82, 0x82, 0x00, 0x00, 0x00, 0x00,
00092 0x00, 0x00, 0x82, 0xc2, 0xa2, 0x92, 0x8a, 0x86, 0x82, 0x82, 0x82, 0x82, 0x00, 0x00, 0x00, 0x00,
00093 0x00, 0x00, 0x7c, 0x82, 0x82, 0x82, 0x82, 0x82, 0x82, 0x82, 0x7c, 0x00, 0x00, 0x00, 0x00,
00094 0x00, 0x00, 0xfc, 0x82, 0x82, 0x82, 0xfc, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00,
00095 0x00, 0x00, 0x7c, 0x82, 0x82, 0x82, 0x82, 0x92, 0x92, 0x92, 0x7c, 0x08, 0x04, 0x00, 0x00, 0x00,
00096 0x00, 0x00, 0xfc, 0x82, 0x82, 0x82, 0xfc, 0x84, 0x82, 0x82, 0x82, 0x82, 0x00, 0x00, 0x00, 0x00,
00097 0x00, 0x00, 0x7c, 0x82, 0x82, 0x80, 0x60, 0x30, 0x0c, 0x02, 0x82, 0x82, 0x7c, 0x00, 0x00, 0x00, 0x00,
00098 0x00, 0x00, 0xfe, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x00, 0x00, 0x00, 0x00,
00099 0x00, 0x00, 0x82, 0x82, 0x82, 0x82, 0x82, 0x82, 0x82, 0x82, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
00100 0x00, 0x00, 0x82, 0x82, 0x82, 0x82, 0x82, 0x82, 0x44, 0x28, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00,
00101 0x00, 0x00, 0x82, 0x82, 0x82, 0x82, 0x82, 0x92, 0x92, 0x92, 0xaa, 0x44, 0x00, 0x00, 0x00, 0x00,
00102 0x00, 0x00, 0x82, 0x82, 0x44, 0x28, 0x10, 0x28, 0x44, 0x82, 0x82, 0x82, 0x00, 0x00, 0x00, 0x00,
00103 0x00, 0x00, 0x82, 0x82, 0x44, 0x28, 0x10, 0x10, 0x10, 0x10, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00,
00104 0x00, 0x00, 0xfe, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x80, 0xfe, 0x00, 0x00, 0x00, 0x00,
00105 0x00, 0x00, 0x38, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00,
00106 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x00, 0x00, 0x00, 0x00,
00107 0x00, 0x00, 0x38, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00,
00108 0x10, 0x28, 0x44, 0x82, 0x00, 0x00,
00109 0x00, 0x00,
00110 0x00, 0x20, 0x20, 0x20, 0x10, 0x00, 0x00,
00111 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x04, 0x04, 0x7c, 0x84, 0x84, 0x84, 0x7e, 0x00, 0x00, 0x00, 0x00,
00112 0x00, 0x00, 0x80, 0x80, 0xbc, 0xc2, 0x82, 0x82, 0x82, 0x82, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00,
00113 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0x82, 0x80, 0x80, 0x80, 0x82, 0x7c, 0x00, 0x00, 0x00, 0x00,
00114 0x00, 0x00, 0x02, 0x02, 0x02, 0x7a, 0x86, 0x82, 0x82, 0x82, 0x7e, 0x00, 0x00, 0x00, 0x00, 0x00,
00115 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0x82, 0x82, 0x82, 0xfe, 0x80, 0x80, 0x7c, 0x00, 0x00, 0x00, 0x00,
00116 0x00, 0x00, 0x1c, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00,
00117 0x00, 0x00, 0x00, 0x00, 0x00, 0x7e, 0x82, 0x82, 0x82, 0x86, 0x7a, 0x02, 0x02, 0x7c, 0x00,
00118 0x00, 0x00, 0x80, 0x80, 0x80, 0xbc, 0xc2, 0x82, 0x82, 0x82, 0x82, 0x82, 0x00, 0x00, 0x00, 0x00,
00119 0x00, 0x00, 0x10, 0x10, 0x00, 0x70, 0x10, 0x10, 0x10, 0x10, 0x10, 0x7c, 0x00, 0x00, 0x00, 0x00,
00120 0x00, 0x00, 0x08, 0x08, 0x00, 0x38, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x70, 0x00,
00121 0x00, 0x00, 0x80, 0x80, 0x86, 0x98, 0x90, 0x88, 0x84, 0x82, 0x00, 0x00, 0x00, 0x00, 0x00,
00122 0x00, 0x00, 0x60, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x1c, 0x00, 0x00, 0x00, 0x00,
00123 0x00, 0x00, 0x00, 0x00, 0x00, 0xec, 0x92, 0x92, 0x92, 0x92, 0x92, 0x00, 0x00, 0x00, 0x00,
00124 0x00, 0x00, 0x00, 0x00, 0x00, 0xbc, 0xc2, 0x82, 0x82, 0x82, 0x82, 0x82, 0x00, 0x00, 0x00, 0x00,
00125 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0x82, 0x82, 0x82, 0x82, 0x82, 0x7c, 0x00, 0x00, 0x00, 0x00,
00126 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xbc, 0xc2, 0x82, 0x82, 0x82, 0x82, 0xbc, 0x80, 0x80, 0x80, 0x00,
00127 0x00, 0x00, 0x00, 0x00, 0x00, 0x7e, 0x82, 0x82, 0x82, 0x82, 0x86, 0x7a, 0x02, 0x02, 0x02, 0x00,
00128 0x00, 0x00, 0x00, 0x00, 0x00, 0xbc, 0xc2, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00,
00129 0x00, 0x00, 0x00, 0x00, 0x00, 0x7e, 0x80, 0x80, 0x80, 0x7c, 0x02, 0x02, 0xfc, 0x00, 0x00, 0x00, 0x00,
00130 0x00, 0x00, 0x00, 0x20, 0x20, 0x20, 0xfc, 0x20, 0x20, 0x20, 0x20, 0x20, 0x1c, 0x00, 0x00, 0x00, 0x00,
00131 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x82, 0x82, 0x82, 0x82, 0x82, 0x86, 0x7a, 0x00, 0x00, 0x00, 0x00,
00132 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x82, 0x82, 0x82, 0x82, 0x82, 0x44, 0x28, 0x10, 0x00, 0x00, 0x00, 0x00,
00133 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x82, 0x82, 0x82, 0x82, 0x92, 0x92, 0x92, 0x6c, 0x00, 0x00, 0x00, 0x00,
00134 0x00, 0x00, 0x00, 0x00, 0x00, 0x82, 0x44, 0x28, 0x10, 0x28, 0x44, 0x82, 0x82, 0x00, 0x00, 0x00, 0x00,
00135 0x00, 0x00, 0x00, 0x00, 0x00, 0x82, 0x82, 0x82, 0x82, 0x82, 0x86, 0x7a, 0x02, 0x02, 0x7c, 0x00,
00136 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x08, 0x10, 0x20, 0x40, 0xfe, 0x00, 0x00, 0x00, 0x00,
00137 0x00, 0x00, 0x0c, 0x10, 0x10, 0x10, 0x60, 0x10, 0x10, 0x10, 0x10, 0x0c, 0x00, 0x00, 0x00, 0x00,
00138 0x00, 0x10, 0x00,
00139 0x00, 0x00, 0x60, 0x10, 0x10, 0x10, 0x0c, 0x10, 0x10, 0x10, 0x10, 0x10, 0x60, 0x00, 0x00, 0x00, 0x00,
00140 0x00, 0x00, 0x62, 0x92, 0x8c, 0x00, 0x00,
00141 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x28, 0x44, 0x82, 0x82, 0x82, 0x00, 0x00, 0x00, 0x00,
00142 0x00, 0x00, 0x7c, 0x82, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x82, 0x82, 0x7c, 0x10, 0x08, 0x70, 0x00,
00143 0x00, 0x00, 0x44, 0x44, 0x00, 0x82, 0x82, 0x82, 0x82, 0x82, 0x82, 0x7a, 0x00, 0x00, 0x00, 0x00,
00144 0x00, 0x08, 0x10, 0x20, 0x00, 0x7c, 0x82, 0x82, 0x82, 0x0fe, 0x80, 0x80, 0x7e, 0x00, 0x00, 0x00, 0x00,
00145 0x00, 0x10, 0x28, 0x44, 0x00, 0x7c, 0x02, 0x02, 0x7e, 0x82, 0x82, 0x7e, 0x00, 0x00, 0x00, 0x00,
00146 0x00, 0x00, 0x44, 0x44, 0x00, 0x7c, 0x02, 0x02, 0x7e, 0x82, 0x82, 0x7e, 0x00, 0x00, 0x00, 0x00,
00147 0x00, 0x20, 0x10, 0x08, 0x00, 0x7c, 0x02, 0x02, 0x7e, 0x82, 0x82, 0x7e, 0x00, 0x00, 0x00, 0x00,
00148 0x00, 0x18, 0x24, 0x18, 0x00, 0x7c, 0x02, 0x02, 0x7e, 0x82, 0x82, 0x7e, 0x00, 0x00, 0x00, 0x00,
00149 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0x82, 0x80, 0x80, 0x80, 0x82, 0x7c, 0x10, 0x08, 0x70, 0x00,
00150 0x00, 0x10, 0x28, 0x44, 0x00, 0x7c, 0x82, 0x82, 0x82, 0x0fe, 0x80, 0x80, 0x7e, 0x00, 0x00, 0x00, 0x00,
00151 0x00, 0x00, 0x44, 0x44, 0x00, 0x7c, 0x82, 0x82, 0x82, 0x0fe, 0x80, 0x80, 0x7e, 0x00, 0x00, 0x00, 0x00,
00152 0x00, 0x20, 0x10, 0x08, 0x00, 0x7c, 0x82, 0x82, 0x82, 0x0fe, 0x80, 0x80, 0x7e, 0x00, 0x00, 0x00, 0x00,
00153 0x00, 0x00, 0x48, 0x48, 0x00, 0x70, 0x10, 0x10, 0x10, 0x10, 0x0c, 0x00, 0x00, 0x00, 0x00,
00154 0x00, 0x10, 0x28, 0x44, 0x00, 0x70, 0x10, 0x10, 0x10, 0x10, 0x10, 0x7c, 0x00, 0x00, 0x00, 0x00,
00155 0x00, 0x40, 0x20, 0x10, 0x00, 0x70, 0x10, 0x10, 0x10, 0x10, 0x10, 0x7c, 0x00, 0x00, 0x00, 0x00,
00156 0x44, 0x44, 0x00, 0x38, 0x44, 0x82, 0x82, 0x82, 0x0fe, 0x82, 0x82, 0x82, 0x00, 0x00, 0x00, 0x00,
00157 0x38, 0x44, 0x44, 0x38, 0x44, 0x82, 0x82, 0x0fe, 0x82, 0x82, 0x82, 0x00, 0x00, 0x00, 0x00, 0x00,
00158 0x0c, 0x30, 0x00, 0xfe, 0x80, 0x80, 0x80, 0xfc, 0x80, 0x80, 0x80, 0x0fe, 0x00, 0x00, 0x00, 0x00,
00159 0x00, 0x00, 0x00, 0x00, 0x00, 0x6c, 0x92, 0x12, 0x7e, 0x90, 0x90, 0x6e, 0x00, 0x00, 0x00, 0x00,
00160 0x00, 0x00, 0x3e, 0x48, 0x88, 0x88, 0x0fe, 0x88, 0x88, 0x88, 0x8e, 0x00, 0x00, 0x00, 0x00,
00161 0x00, 0x10, 0x28, 0x44, 0x00, 0x7c, 0x82, 0x82, 0x82, 0x82, 0x7c, 0x00, 0x00, 0x00, 0x00,
00162 0x00, 0x00, 0x44, 0x44, 0x00, 0x7c, 0x82, 0x82, 0x82, 0x82, 0x82, 0x7c, 0x00, 0x00, 0x00, 0x00,
00163 0x00, 0x20, 0x10, 0x08, 0x00, 0x7c, 0x82, 0x82, 0x82, 0x82, 0x82, 0x7c, 0x00, 0x00, 0x00, 0x00,
00164 0x00, 0x10, 0x28, 0x44, 0x00, 0x82, 0x82, 0x82, 0x82, 0x82, 0x86, 0x7a, 0x00, 0x00, 0x00, 0x00,
00165 0x00, 0x20, 0x10, 0x08, 0x00, 0x82, 0x82, 0x82, 0x82, 0x82, 0x86, 0x7a, 0x00, 0x00, 0x00, 0x00,
00166 0x00, 0x00, 0x44, 0x44, 0x00, 0x82, 0x82, 0x82, 0x82, 0x82, 0x86, 0x7a, 0x02, 0x02, 0x7c, 0x00,
00167 0x44, 0x44, 0x00, 0x7c, 0x82, 0x82, 0x82, 0x82, 0x82, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
00168 0x44, 0x44, 0x00, 0x82, 0x82, 0x82, 0x82, 0x82, 0x82, 0x82, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
00169 0x00, 0x00, 0x10, 0x08, 0x7c, 0x82, 0x80, 0x80, 0x80, 0x82, 0x7c, 0x10, 0x08, 0x72, 0xcc, 0x00,
00170 0x00, 0x00, 0x38, 0x44, 0x40, 0x40, 0x0fe, 0x40, 0x40, 0x72, 0xcc, 0x00, 0x00, 0x00, 0x00, 0x00,
00171 0x00, 0x00, 0x82, 0x82, 0x44, 0x28, 0x10, 0x7c, 0x10, 0x7c, 0x10, 0x10, 0x00, 0x00, 0x00, 0x00,
00172 0x00, 0x00, 0xf8, 0x84, 0x84, 0xf8, 0x80, 0x84, 0x88, 0x88, 0x84, 0x00, 0x00, 0x00,
```



```

00260 0x00, 0x00, 0x00, 0x00, 0x10, 0x00, 0xfe, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00,
00261 0x00, 0x00, 0x00, 0x00, 0x64, 0x98, 0x00, 0x64, 0x98, 0x00, 0x00, 0x00, 0x00, 0x00,
00262 0x00, 0x70, 0x88, 0x88, 0x70, 0x00, 0x00, 0x00, 0x30, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00,
00263 0x00, 0x00,
00264 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00265 0x00, 0x0e, 0x08, 0x08, 0x08, 0x08, 0x08, 0xc8, 0x28, 0x18, 0x08, 0x00, 0x00, 0x00, 0x00,
00266 0x00, 0xb8, 0x44, 0x44, 0x44, 0x44, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00267 0x00, 0x60, 0x90, 0x10, 0x20, 0x40, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00268 0x00, 0x00, 0x00, 0x00, 0x7c, 0x7c, 0x7c, 0x7c, 0x7c, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
00269 0x00, 0x00
00270 };
00271 static void fbterm_save_state(struct term_context *_ctx) {
00272     struct fbterm_context *ctx = (void *)_ctx;
00273     ctx->saved_state_text_fg = ctx->text_fg;
00274     ctx->saved_state_text_bg = ctx->text_bg;
00275     ctx->saved_state_cursor_x = ctx->cursor_x;
00276     ctx->saved_state_cursor_y = ctx->cursor_y;
00277 }
00278
00279 static void fbterm_restore_state(struct term_context *_ctx) {
00280     struct fbterm_context *ctx = (void *)_ctx;
00281     ctx->text_fg = ctx->saved_state_text_fg;
00282     ctx->text_bg = ctx->saved_state_text_bg;
00283     ctx->cursor_x = ctx->saved_state_cursor_x;
00284     ctx->cursor_y = ctx->saved_state_cursor_y;
00285 }
00286
00287 static void fbterm_swap_palette(struct term_context *_ctx) {
00288     struct fbterm_context *ctx = (void *)_ctx;
00289     uint32_t tmp = ctx->text_bg;
00290     ctx->text_bg = ctx->text_fg;
00291     ctx->text_fg = tmp;
00292 }
00293
00294 static void plot_char(struct term_context *_ctx, struct fbterm_char *c, size_t x, size_t y) {
00295     struct fbterm_context *ctx = (void *)_ctx;
00296
00297     if (x >= _ctx->cols || y >= _ctx->rows) {
00298         return;
00299     }
00300
00301     x = ctx->offset_x + x * ctx->glyph_width;
00302     y = ctx->offset_y + y * ctx->glyph_height;
00303
00304     bool *glyph = &ctx->font_bool[c->c * ctx->font_height * ctx->font_width];
00305     // naming: fx,fy for font coordinates, gx,gy for glyph coordinates
00306     for (size_t gy = 0; gy < ctx->glyph_height; gy++) {
00307         uint8_t fy = gy / ctx->font_scale_y;
00308         volatile uint32_t *fb_line = ctx->framebuffer + x + (y + gy) * (ctx->pitch / 4);
00309         uint32_t *canvas_line = ctx->canvas + x + (y + gy) * ctx->width;
00310         for (size_t fx = 0; fx < ctx->font_width; fx++) {
00311             bool draw = glyph[fy * ctx->font_width + fx];
00312             for (size_t i = 0; i < ctx->font_scale_x; i++) {
00313                 size_t gx = ctx->font_scale_x * fx + i;
00314                 uint32_t bg = c->bg == 0xffffffff ? canvas_line[gx] : c->bg;
00315                 uint32_t fg = c->fg == 0xffffffff ? canvas_line[gx] : c->fg;
00316                 fb_line[gx] = draw ? fg : bg;
00317             }
00318         }
00319     }
00320 }
00321
00322 static void plot_char_fast(struct term_context *_ctx, struct fbterm_char *old, struct fbterm_char *c,
00323     size_t x, size_t y) {
00324     struct fbterm_context *ctx = (void *)_ctx;
00325
00326     if (x >= _ctx->cols || y >= _ctx->rows) {
00327         return;
00328     }
00329
00330     x = ctx->offset_x + x * ctx->glyph_width;
00331     y = ctx->offset_y + y * ctx->glyph_height;
00332
00333     bool *new_glyph = &ctx->font_bool[c->c * ctx->font_height * ctx->font_width];
00334     bool *old_glyph = &ctx->font_bool[old->c * ctx->font_height * ctx->font_width];
00335     for (size_t gy = 0; gy < ctx->glyph_height; gy++) {
00336         uint8_t fy = gy / ctx->font_scale_y;
00337         volatile uint32_t *fb_line = ctx->framebuffer + x + (y + gy) * (ctx->pitch / 4);
00338         uint32_t *canvas_line = ctx->canvas + x + (y + gy) * ctx->width;
00339         for (size_t fx = 0; fx < ctx->font_width; fx++) {
00340             bool old_draw = old_glyph[fy * ctx->font_width + fx];
00341             bool new_draw = new_glyph[fy * ctx->font_width + fx];
00342             if (old_draw == new_draw)
00343                 continue;
00344             for (size_t i = 0; i < ctx->font_scale_x; i++) {
00345                 size_t gx = ctx->font_scale_x * fx + i;
00346                 uint32_t bg = c->bg == 0xffffffff ? canvas_line[gx] : c->bg;
00347             }
00348         }
00349     }
00350 }

```

```

00346             uint32_t fg = c->fg == 0xffffffff ? canvas_line[gx] : c->fg;
00347         }
00348     }
00349 }
00350 }
00351 }
00352
00353 static inline bool compare_char(struct fbterm_char *a, struct fbterm_char *b) {
00354     return !(a->c != b->c || a->bg != b->bg || a->fg != b->fg);
00355 }
00356
00357 static void push_to_queue(struct term_context *_ctx, struct fbterm_char *c, size_t x, size_t y) {
00358     struct fbterm_context *ctx = (void *)_ctx;
00359
00360     if (x >= _ctx->cols || y >= _ctx->rows) {
00361         return;
00362     }
00363
00364     size_t i = y * _ctx->cols + x;
00365
00366     struct fbterm_queue_item *q = ctx->map[i];
00367
00368     if (q == NULL) {
00369         if (compare_char(&ctx->grid[i], c)) {
00370             return;
00371         }
00372         q = &ctx->queue[ctx->queue_i++];
00373         q->x = x;
00374         q->y = y;
00375         ctx->map[i] = q;
00376     }
00377
00378     q->c = *c;
00379 }
00380
00381 static void fbterm_revscroll(struct term_context *_ctx) {
00382     struct fbterm_context *ctx = (void *)_ctx;
00383
00384     for (size_t i = (_ctx->scroll_bottom_margin - 1) * _ctx->cols - 1; ; i--) {
00385         struct fbterm_char *c;
00386         struct fbterm_queue_item *q = ctx->map[i];
00387         if (q != NULL) {
00388             c = &q->c;
00389         } else {
00390             c = &ctx->grid[i];
00391         }
00392         push_to_queue(_ctx, c, (i + _ctx->cols) % _ctx->cols, (i + _ctx->cols) / _ctx->cols);
00393         if (i == _ctx->scroll_top_margin * _ctx->cols) {
00394             break;
00395         }
00396     }
00397
00398     // Clear the first line of the screen.
00399     struct fbterm_char empty;
00400     empty.c = ' ';
00401     empty.fg = ctx->text_fg;
00402     empty.bg = ctx->text_bg;
00403     for (size_t i = _ctx->scroll_top_margin * _ctx->cols;
00404         i < (_ctx->scroll_top_margin + 1) * _ctx->cols; i++) {
00405         push_to_queue(_ctx, &empty, i % _ctx->cols, i / _ctx->cols);
00406     }
00407 }
00408
00409 static void fbterm_scroll(struct term_context *_ctx) {
00410     struct fbterm_context *ctx = (void *)_ctx;
00411
00412     for (size_t i = (_ctx->scroll_top_margin + 1) * _ctx->cols;
00413         i < _ctx->scroll_bottom_margin * _ctx->cols; i++) {
00414         struct fbterm_char *c;
00415         struct fbterm_queue_item *q = ctx->map[i];
00416         if (q != NULL) {
00417             c = &q->c;
00418         } else {
00419             c = &ctx->grid[i];
00420         }
00421         push_to_queue(_ctx, c, (i - _ctx->cols) % _ctx->cols, (i - _ctx->cols) / _ctx->cols);
00422     }
00423
00424     // Clear the last line of the screen.
00425     struct fbterm_char empty;
00426     empty.c = ' ';
00427     empty.fg = ctx->text_fg;
00428     empty.bg = ctx->text_bg;
00429     for (size_t i = (_ctx->scroll_bottom_margin - 1) * _ctx->cols;
00430         i < _ctx->scroll_bottom_margin * _ctx->cols; i++) {
00431         push_to_queue(_ctx, &empty, i % _ctx->cols, i / _ctx->cols);
00432     }

```

```

00433 }
00434
00435 static void fbterm_clear(struct term_context *_ctx, bool move) {
00436     struct fbterm_context *ctx = (void *)_ctx;
00437
00438     struct fbterm_char empty;
00439     empty.c = ' ';
00440     empty.fg = ctx->text_fg;
00441     empty.bg = ctx->text_bg;
00442     for (size_t i = 0; i < _ctx->rows * _ctx->cols; i++) {
00443         push_to_queue(_ctx, &empty, i % _ctx->cols, i / _ctx->cols);
00444     }
00445
00446     if (move) {
00447         ctx->cursor_x = 0;
00448         ctx->cursor_y = 0;
00449     }
00450 }
00451
00452 static void fbterm_enable_cursor(struct term_context *_ctx) {
00453     struct fbterm_context *ctx = (void *)_ctx;
00454
00455     ctx->cursor_status = true;
00456 }
00457
00458 static bool fbterm_disable_cursor(struct term_context *_ctx) {
00459     struct fbterm_context *ctx = (void *)_ctx;
00460
00461     bool ret = ctx->cursor_status;
00462     ctx->cursor_status = false;
00463     return ret;
00464 }
00465
00466 static void fbterm_set_cursor_pos(struct term_context *_ctx, size_t x, size_t y) {
00467     struct fbterm_context *ctx = (void *)_ctx;
00468
00469     if (x >= _ctx->cols) {
00470         if ((int)x < 0) {
00471             x = 0;
00472         } else {
00473             x = _ctx->cols - 1;
00474         }
00475     }
00476     if (y >= _ctx->rows) {
00477         if ((int)y < 0) {
00478             y = 0;
00479         } else {
00480             y = _ctx->rows - 1;
00481         }
00482     }
00483     ctx->cursor_x = x;
00484     ctx->cursor_y = y;
00485 }
00486
00487 static void fbterm_get_cursor_pos(struct term_context *_ctx, size_t *x, size_t *y) {
00488     struct fbterm_context *ctx = (void *)_ctx;
00489
00490     *x = ctx->cursor_x;
00491     *y = ctx->cursor_y;
00492 }
00493
00494 static void fbterm_move_character(struct term_context *_ctx, size_t new_x, size_t new_y, size_t old_x,
00495                                     size_t old_y) {
00496     struct fbterm_context *ctx = (void *)_ctx;
00497
00498     if (old_x >= _ctx->cols || old_y >= _ctx->rows
00499         || new_x >= _ctx->cols || new_y >= _ctx->rows) {
00500         return;
00501     }
00502
00503     size_t i = old_x + old_y * _ctx->cols;
00504
00505     struct fbterm_char *c;
00506     struct fbterm_queue_item *q = ctx->map[i];
00507     if (q != NULL) {
00508         c = &q->c;
00509     } else {
00510         c = &ctx->grid[i];
00511     }
00512
00513     push_to_queue(_ctx, c, new_x, new_y);
00514
00515 static void fbterm_set_text_fg(struct term_context *_ctx, size_t fg) {
00516     struct fbterm_context *ctx = (void *)_ctx;
00517
00518     ctx->text_fg = ctx->ansi_colours[fg];

```

```
00519 }
00520
00521 static void fbterm_set_text_bg(struct term_context *_ctx, size_t bg) {
00522     struct fbterm_context *ctx = (void *)_ctx;
00523
00524     ctx->text_bg = ctx->ansi_colours[bg];
00525 }
00526
00527 static void fbterm_set_text_fg_bright(struct term_context *_ctx, size_t fg) {
00528     struct fbterm_context *ctx = (void *)_ctx;
00529
00530     ctx->text_fg = ctx->ansi_bright_colours[fg];
00531 }
00532
00533 static void fbterm_set_text_bg_bright(struct term_context *_ctx, size_t bg) {
00534     struct fbterm_context *ctx = (void *)_ctx;
00535
00536     ctx->text_bg = ctx->ansi_bright_colours[bg];
00537 }
00538
00539 static void fbterm_set_text_fg_rgb(struct term_context *_ctx, uint32_t fg) {
00540     struct fbterm_context *ctx = (void *)_ctx;
00541
00542     ctx->text_fg = fg;
00543 }
00544
00545 static void fbterm_set_text_bg_rgb(struct term_context *_ctx, uint32_t bg) {
00546     struct fbterm_context *ctx = (void *)_ctx;
00547
00548     ctx->text_bg = bg;
00549 }
00550
00551 static void fbterm_set_text_fg_default(struct term_context *_ctx) {
00552     struct fbterm_context *ctx = (void *)_ctx;
00553
00554     ctx->text_fg = ctx->default_fg;
00555 }
00556
00557 static void fbterm_set_text_bg_default(struct term_context *_ctx) {
00558     struct fbterm_context *ctx = (void *)_ctx;
00559
00560     ctx->text_bg = 0xffffffff;
00561 }
00562
00563 static void draw_cursor(struct term_context *_ctx) {
00564     struct fbterm_context *ctx = (void *)_ctx;
00565
00566     size_t i = ctx->cursor_x + ctx->cursor_y * _ctx->cols;
00567     struct fbterm_char c;
00568     struct fbterm_queue_item *q = ctx->map[i];
00569     if (q != NULL) {
00570         c = q->c;
00571     } else {
00572         c = ctx->grid[i];
00573     }
00574     uint32_t tmp = c.fg;
00575     c.fg = c.bg;
00576     c.bg = tmp;
00577     plot_char(_ctx, &c, ctx->cursor_x, ctx->cursor_y);
00578     if (q != NULL) {
00579         ctx->grid[i] = q->c;
00580         ctx->map[i] = NULL;
00581     }
00582 }
00583
00584 static void fbterm_double_buffer_flush(struct term_context *_ctx) {
00585     struct fbterm_context *ctx = (void *)_ctx;
00586
00587     if (ctx->cursor_status) {
00588         draw_cursor(_ctx);
00589     }
00590
00591     for (size_t i = 0; i < ctx->queue_i; i++) {
00592         struct fbterm_queue_item *q = &ctx->queue[i];
00593         size_t offset = q->y * _ctx->cols + q->x;
00594         if (ctx->map[offset] == NULL) {
00595             continue;
00596         }
00597         struct fbterm_char *old = &ctx->grid[offset];
00598         if (q->c.bg == old->bg && q->c.fg == old->fg) {
00599             plot_char_fast(_ctx, old, &q->c, q->x, q->y);
00600         } else {
00601             plot_char(_ctx, &q->c, q->x, q->y);
00602         }
00603         ctx->grid[offset] = q->c;
00604         ctx->map[offset] = NULL;
00605     }
```

```

00606
00607     if ((ctx->old_cursor_x != ctx->cursor_x || ctx->old_cursor_y != ctx->cursor_y) ||
00608         ctx->cursor_status == false) {
00609         plot_char(_ctx, &ctx->grid[ctx->old_cursor_x + ctx->old_cursor_y * _ctx->cols],
00610         ctx->old_cursor_x, ctx->old_cursor_y);
00611     }
00612     ctx->old_cursor_x = ctx->cursor_x;
00613     ctx->old_cursor_y = ctx->cursor_y;
00614     ctx->queue_i = 0;
00615 }
00616
00617 static void fbterm_raw_putchar(struct term_context *_ctx, uint8_t c) {
00618     struct fbterm_context *ctx = (void *)_ctx;
00619
00620     struct fbterm_char ch;
00621     ch.c = c;
00622     ch.fg = ctx->text_fg;
00623     ch.bg = ctx->text_bg;
00624     push_to_queue(_ctx, &ch, ctx->cursor_x++, ctx->cursor_y);
00625     if (ctx->cursor_x == _ctx->cols && (ctx->cursor_y < _ctx->scroll_bottom_margin - 1 ||
00626         _ctx->scroll_enabled)) {
00627         ctx->cursor_x = 0;
00628         ctx->cursor_y++;
00629     }
00630     if (ctx->cursor_y == _ctx->scroll_bottom_margin) {
00631         ctx->cursor_y--;
00632         fbterm_scroll(_ctx);
00633     }
00634 }
00635 static void fbterm_full_refresh(struct term_context *_ctx) {
00636     struct fbterm_context *ctx = (void *)_ctx;
00637
00638     for (size_t y = 0; y < ctx->height; y++) {
00639         for (size_t x = 0; x < ctx->width; x++) {
00640             ctx->framebuffer[y * (ctx->pitch / sizeof(uint32_t)) + x] = ctx->canvas[y * ctx->width +
00641             x];
00642         }
00643     }
00644     for (size_t i = 0; i < (size_t)_ctx->rows * _ctx->cols; i++) {
00645         size_t x = i % _ctx->cols;
00646         size_t y = i / _ctx->cols;
00647
00648         plot_char(_ctx, &ctx->grid[i], x, y);
00649     }
00650     if (ctx->cursor_status) {
00651         draw_cursor(_ctx);
00652     }
00653 }
00654 }
00655
00656 static void fbterm_deinit(struct term_context *_ctx, void (*_free)(void *, size_t)) {
00657     struct fbterm_context *ctx = (void *)_ctx;
00658
00659     _free(ctx->font_bits, ctx->font_bits_size);
00660     _free(ctx->font_bool, ctx->font_bool_size);
00661     _free(ctx->grid, ctx->grid_size);
00662     _free(ctx->queue, ctx->queue_size);
00663     _free(ctx->map, ctx->map_size);
00664     _free(ctx->canvas, ctx->canvas_size);
00665     _free(ctx, sizeof(struct fbterm_context));
00666 }
00667
00668 struct term_context *fbterm_init(
00669     void *(*_malloc)(size_t),
00670     uint32_t *framebuffer, size_t width, size_t height, size_t pitch,
00671     uint32_t *canvas,
00672     uint32_t *ansi_colours, uint32_t *ansi_bright_colours,
00673     uint32_t *default_bg, uint32_t *default_fg,
00674     void *font, size_t font_width, size_t font_height, size_t font_spacing,
00675     size_t font_scale_x, size_t font_scale_y,
00676     size_t margin
00677 ) {
00678     struct fbterm_context *ctx = _malloc(sizeof(struct fbterm_context));
00679
00680     struct term_context *_ctx = (void *)ctx;
00681
00682     memset(ctx, 0, sizeof(struct fbterm_context));
00683
00684     ctx->cursor_status = true;
00685
00686     if (ansi_colours != NULL) {
00687         memcpy(ctx->ansi_colours, ansi_colours, sizeof(ctx->ansi_colours));
00688     } else {

```

```

00689     ctx->ansi_colours[0] = 0x00000000; // black
00690     ctx->ansi_colours[1] = 0x00aa0000; // red
00691     ctx->ansi_colours[2] = 0x0000aa00; // green
00692     ctx->ansi_colours[3] = 0x00aa5500; // brown
00693     ctx->ansi_colours[4] = 0x000000aa; // blue
00694     ctx->ansi_colours[5] = 0x00aa00aa; // magenta
00695     ctx->ansi_colours[6] = 0x0000aaaa; // cyan
00696     ctx->ansi_colours[7] = 0x00aaaaaa; // grey
00697 }
00698
00699 if (ansi_bright_colours != NULL) {
00700     memcpy(ctx->ansi_bright_colours, ansi_bright_colours, sizeof(ctx->ansi_bright_colours));
00701 } else {
00702     ctx->ansi_bright_colours[0] = 0x00555555; // black
00703     ctx->ansi_bright_colours[1] = 0x00ff5555; // red
00704     ctx->ansi_bright_colours[2] = 0x0055ff55; // green
00705     ctx->ansi_bright_colours[3] = 0x00ffff55; // brown
00706     ctx->ansi_bright_colours[4] = 0x005555ff; // blue
00707     ctx->ansi_bright_colours[5] = 0x00ff55ff; // magenta
00708     ctx->ansi_bright_colours[6] = 0x0055ffff; // cyan
00709     ctx->ansi_bright_colours[7] = 0x00ffffff; // grey
00710 }
00711
00712 if (default_bg != NULL) {
00713     ctx->default_bg = *default_bg;
00714 } else {
00715     ctx->default_bg = 0x00000000; // background (black)
00716 }
00717
00718 if (default_fg != NULL) {
00719     ctx->default_fg = *default_fg;
00720 } else {
00721     ctx->default_fg = 0x00aaaaaa; // foreground (grey)
00722 }
00723
00724 ctx->text_fg = ctx->default_fg;
00725 ctx->text_bg = 0xffffffff;
00726
00727 ctx->framebuffer = (void *)framebuffer;
00728 ctx->width = width;
00729 ctx->height = height;
00730 ctx->pitch = pitch;
00731
00732 #define FONT_BYTES ((font_width * font_height * FBTERM_FONT_GLYPHS) / 8)
00733
00734 if (font != NULL) {
00735     ctx->font_width = font_width;
00736     ctx->font_height = font_height;
00737     ctx->font_bits = _malloc(FONT_BYTES);
00738     memcpy(ctx->font_bits, font, FONT_BYTES);
00739 } else {
00740     ctx->font_width = font_width = 8;
00741     ctx->font_height = font_height = 16;
00742     font_spacing = 1;
00743     ctx->font_bits = _malloc(FONT_BYTES);
00744     memcpy(ctx->font_bits, builtin_font, FONT_BYTES);
00745 }
00746
00747 ctx->font_bits_size = FONT_BYTES;
00748
00749 #undef FONT_BYTES
00750
00751     ctx->font_width += font_spacing;
00752
00753     ctx->font_bool_size = FBTERM_FONT_GLYPHS * font_height * ctx->font_width * sizeof(bool);
00754     ctx->font_bool = _malloc(ctx->font_bool_size);
00755
00756     for (size_t i = 0; i < FBTERM_FONT_GLYPHS; i++) {
00757         uint8_t *glyph = &ctx->font_bits[i * font_height];
00758
00759         for (size_t y = 0; y < font_height; y++) {
00760             // NOTE: the characters in VGA fonts are always one byte wide.
00761             // 9 dot wide fonts have 8 dots and one empty column, except
00762             // characters 0xC0-0xDF replicate column 9.
00763             for (size_t x = 0; x < 8; x++) {
00764                 size_t offset = i * font_height * ctx->font_width + y * ctx->font_width + x;
00765
00766                 if ((glyph[y] & (0x80 >> x))) {
00767                     ctx->font_bool[offset] = true;
00768                 } else {
00769                     ctx->font_bool[offset] = false;
00770                 }
00771             }
00772             // fill columns above 8 like VGA Line Graphics Mode does
00773             for (size_t x = 8; x < ctx->font_width; x++) {
00774                 size_t offset = i * font_height * ctx->font_width + y * ctx->font_width + x;
00775

```

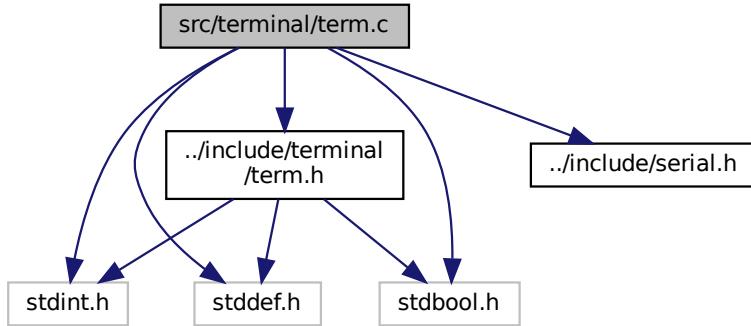
```

00776             if (i >= 0xc0 && i <= 0xdf) {
00777                 ctx->font_bool[offset] = (glyph[y] & 1);
00778             } else {
00779                 ctx->font_bool[offset] = false;
00780             }
00781         }
00782     }
00783 }
00784
00785     ctx->font_scale_x = font_scale_x;
00786     ctx->font_scale_y = font_scale_y;
00787
00788     ctx->glyph_width = ctx->font_width * font_scale_x;
00789     ctx->glyph_height = font_height * font_scale_y;
00790
00791     _ctx->cols = (ctx->width - margin * 2) / ctx->glyph_width;
00792     _ctx->rows = (ctx->height - margin * 2) / ctx->glyph_height;
00793
00794     ctx->offset_x = margin + ((ctx->width - margin * 2) % ctx->glyph_width) / 2;
00795     ctx->offset_y = margin + ((ctx->height - margin * 2) % ctx->glyph_height) / 2;
00796
00797     ctx->grid_size = _ctx->rows * _ctx->cols * sizeof(struct fbterm_char);
00798     ctx->grid = _malloc(ctx->grid_size);
00799     for (size_t i = 0; i < _ctx->rows * _ctx->cols; i++) {
00800         ctx->grid[i].c = ' ';
00801         ctx->grid[i].fg = ctx->text_fg;
00802         ctx->grid[i].bg = ctx->text_bg;
00803     }
00804
00805     ctx->queue_size = _ctx->rows * _ctx->cols * sizeof(struct fbterm_queue_item);
00806     ctx->queue = _malloc(ctx->queue_size);
00807     ctx->queue_i = 0;
00808
00809     ctx->map_size = _ctx->rows * _ctx->cols * sizeof(struct fbterm_queue_item *);
00810     ctx->map = _malloc(ctx->map_size);
00811
00812     ctx->canvas_size = ctx->width * ctx->height * sizeof(uint32_t);
00813     ctx->canvas = _malloc(ctx->canvas_size);
00814     if (canvas != NULL) {
00815         memcpy(ctx->canvas, canvas, ctx->canvas_size);
00816     } else {
00817         for (size_t i = 0; i < ctx->width * ctx->height; i++) {
00818             ctx->canvas[i] = ctx->default_bg;
00819         }
00820     }
00821
00822     _ctx->raw_putchar = fbterm_raw_putchar;
00823     _ctx->clear = fbterm_clear;
00824     _ctx->enable_cursor = fbterm_enable_cursor;
00825     _ctx->disable_cursor = fbterm_disable_cursor;
00826     _ctx->set_cursor_pos = fbterm_set_cursor_pos;
00827     _ctx->get_cursor_pos = fbterm_get_cursor_pos;
00828     _ctx->set_text_fg = fbterm_set_text_fg;
00829     _ctx->set_text_bg = fbterm_set_text_bg;
00830     _ctx->set_text_fg_bright = fbterm_set_text_fg_bright;
00831     _ctx->set_text_bg_bright = fbterm_set_text_bg_bright;
00832     _ctx->set_text_fg_rgb = fbterm_set_text_fg_rgb;
00833     _ctx->set_text_bg_rgb = fbterm_set_text_bg_rgb;
00834     _ctx->set_text_fg_default = fbterm_set_text_fg_default;
00835     _ctx->set_text_bg_default = fbterm_set_text_bg_default;
00836     _ctx->move_character = fbterm_move_character;
00837     _ctx->scroll = fbterm_scroll;
00838     _ctx->revscroll = fbterm_revscroll;
00839     _ctx->swap_palette = fbterm_swap_palette;
00840     _ctx->save_state = fbterm_save_state;
00841     _ctx->restore_state = fbterm_restore_state;
00842     _ctx->double_buffer_flush = fbterm_double_buffer_flush;
00843     _ctx->full_refresh = fbterm_full_refresh;
00844     _ctx->deinit = fbterm_deinit;
00845
00846     term_context_reinit(_ctx);
00847
00848     return _ctx;
00849 }
```

## 4.137 src/terminal/term.c File Reference

```
#include <stdint.h>
#include <stddef.h>
#include <stdbool.h>
```

```
#include "../include/terminal/term.h"
#include "../include/serial.h"
Include dependency graph for term.c:
```



## Macros

- `#define CHARSET_DEFAULT 0`
- `#define CHARSET_DEC_SPECIAL 1`

## Functions

- `void term_context_reinit (struct term_context *ctx)`
- `static void term_putchar (struct term_context *ctx, uint8_t c)`
- `void term_write (struct term_context *ctx, const char *buf, size_t count)`
- `static void sgr (struct term_context *ctx)`
- `static void dec_private_parse (struct term_context *ctx, uint8_t c)`
- `static void linux_private_parse (struct term_context *ctx)`
- `static void mode_toggle (struct term_context *ctx, uint8_t c)`
- `static void control_sequence_parse (struct term_context *ctx, uint8_t c)`
- `static void restore_state (struct term_context *ctx)`
- `static void save_state (struct term_context *ctx)`
- `static void escape_parse (struct term_context *ctx, uint8_t c)`
- `static uint8_t dec_special_to_cp437 (uint8_t c)`

## Variables

- `static const uint32_t col256 []`

### 4.137.1 Macro Definition Documentation

#### 4.137.1.1 CHARSET\_DEC\_SPECIAL

```
#define CHARSET_DEC_SPECIAL 1
```

Definition at line [44](#) of file [term.c](#).

#### 4.137.1.2 CHARSET\_DEFAULT

```
#define CHARSET_DEFAULT 0
```

Definition at line [43](#) of file [term.c](#).

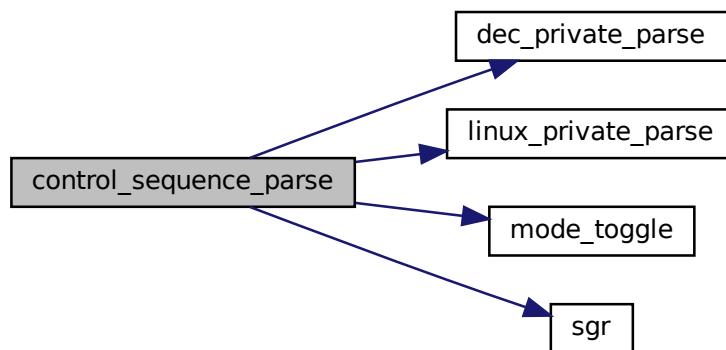
### 4.137.2 Function Documentation

#### 4.137.2.1 control\_sequence\_parse()

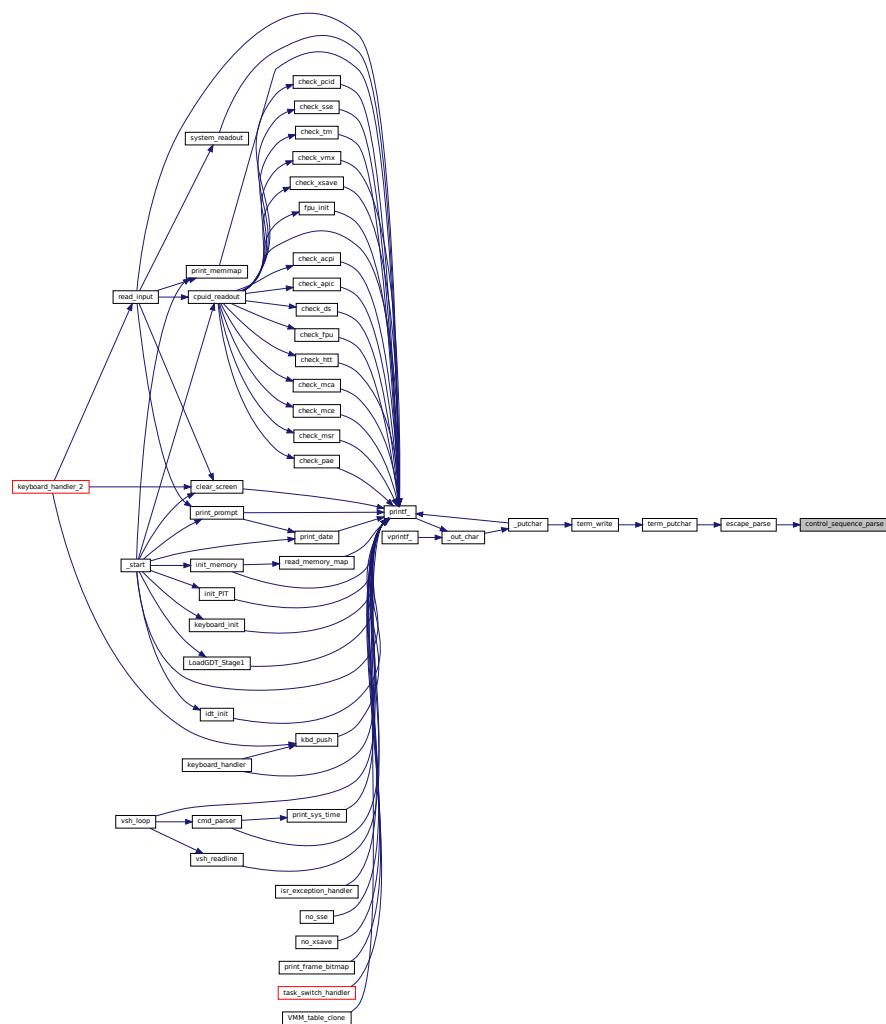
```
static void control_sequence_parse (
    struct term_context * ctx,
    uint8_t c ) [static]
```

Definition at line [436](#) of file [term.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

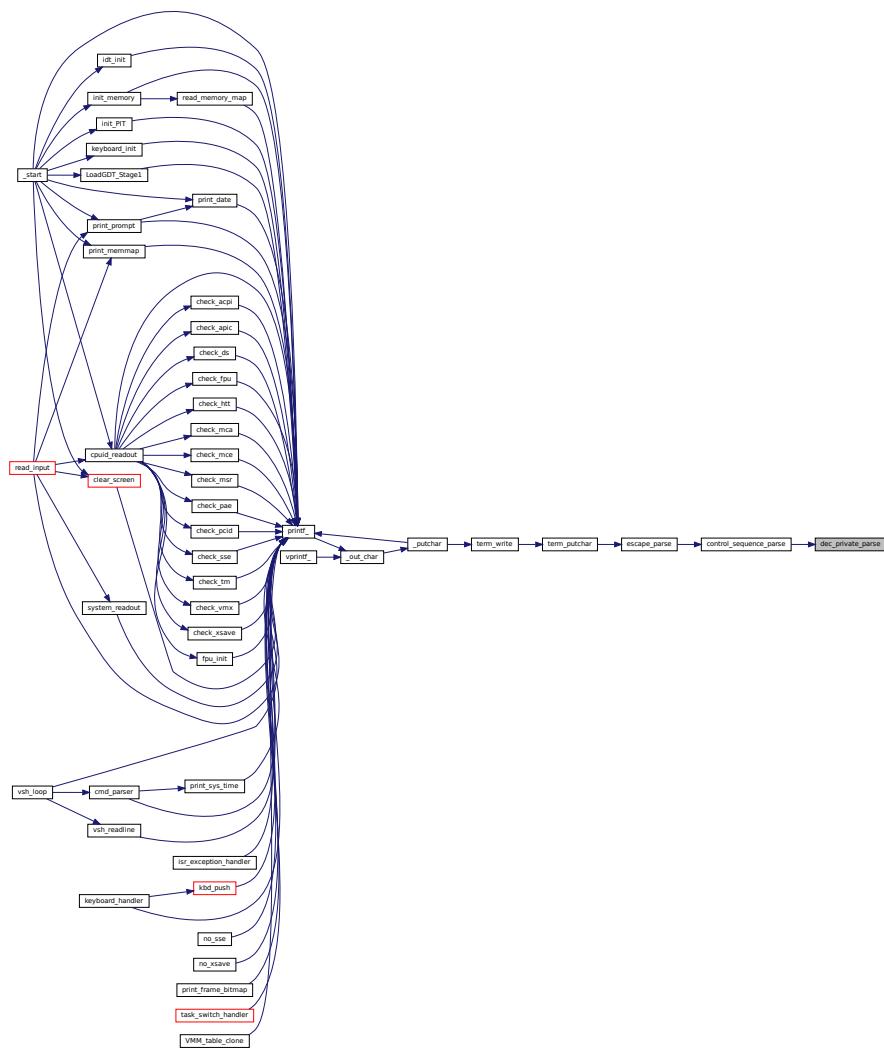


#### 4.137.2.2 dec\_private\_parse()

```
static void dec_private_parse (
    struct term_context * ctx,
    uint8_t c ) [static]
```

Definition at line 344 of file [term.c](#).

Here is the caller graph for this function:

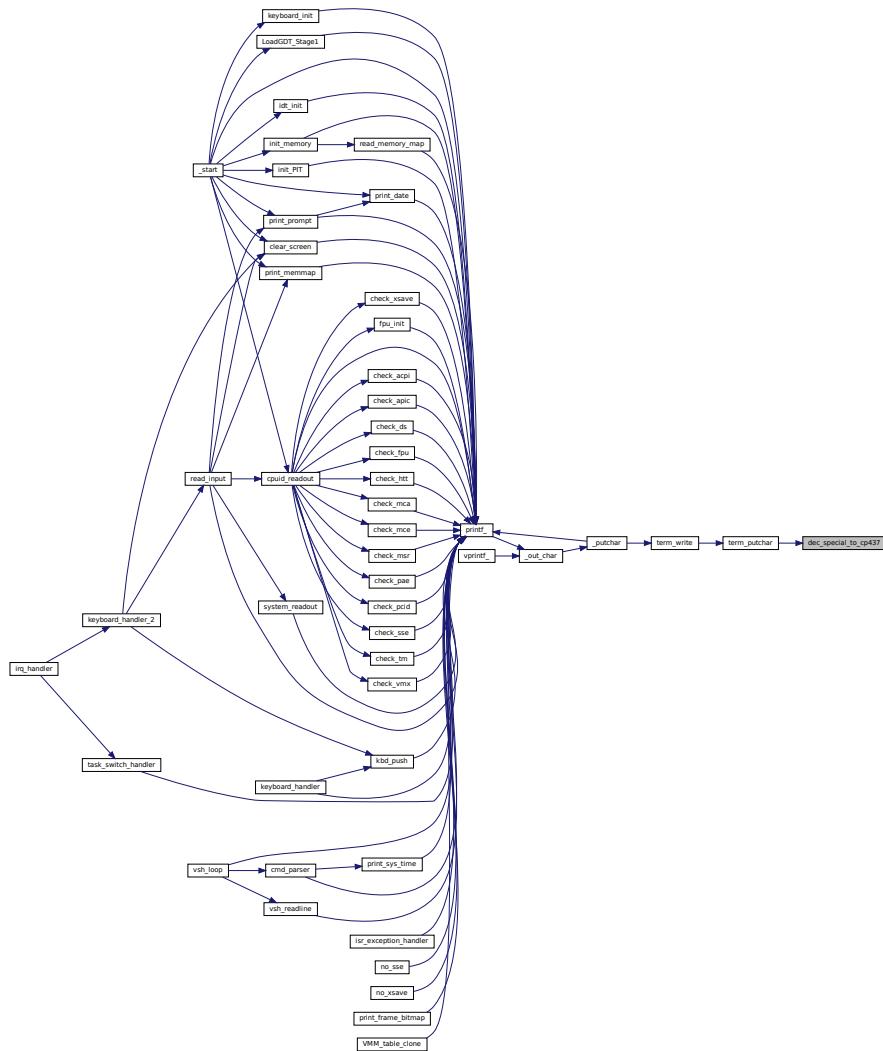


#### 4.137.2.3 dec\_special\_to\_cp437()

```
static uint8_t dec_special_to_cp437 (
    uint8_t c ) [static]
```

Definition at line 867 of file [term.c](#).

Here is the caller graph for this function:

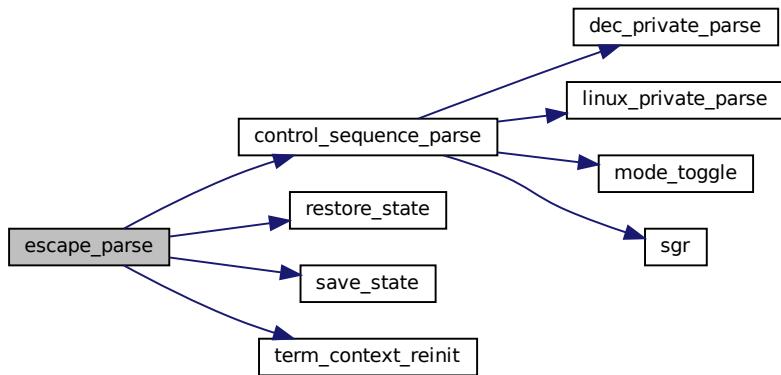


#### 4.137.2.4 escape\_parse()

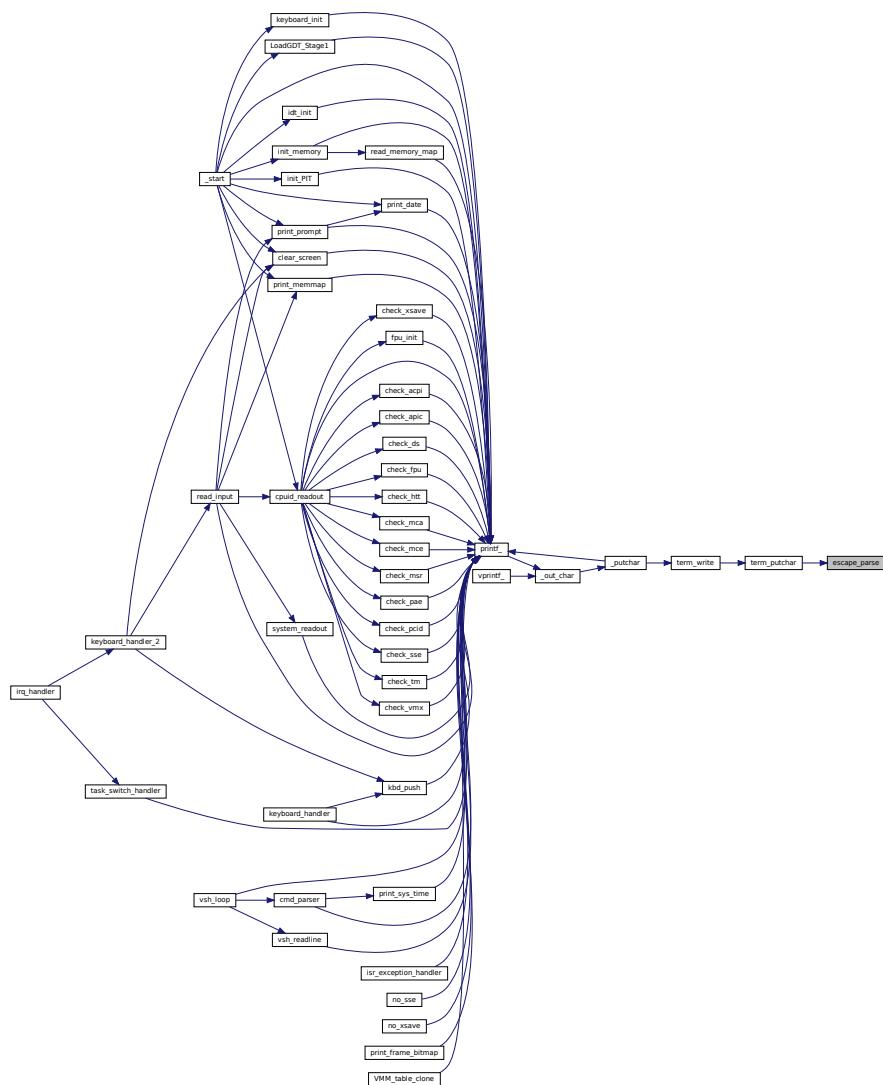
```
static void escape_parse (
    struct term_context * ctx,
    uint8_t c ) [static]
```

Definition at line 773 of file [term.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

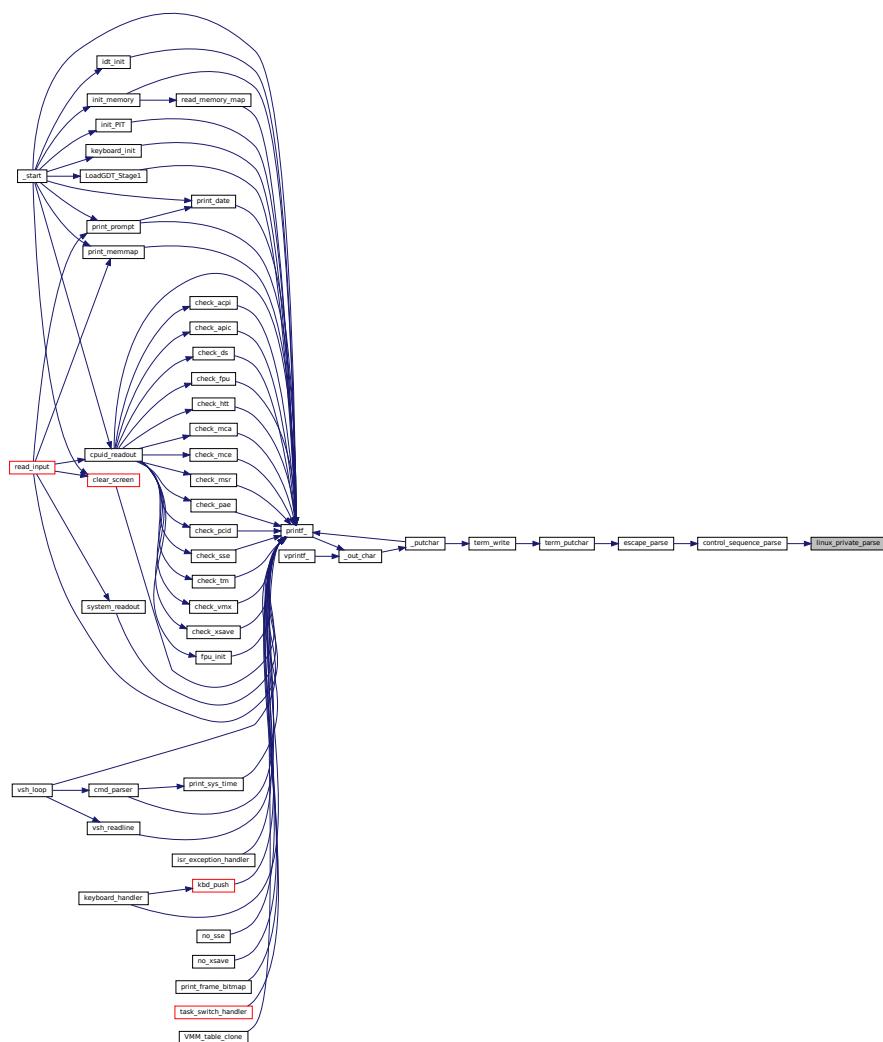


#### 4.137.2.5 linux\_private\_parse()

```
static void linux_private_parse (
    struct term_context * ctx ) [static]
```

Definition at line 389 of file [term.c](#).

Here is the caller graph for this function:

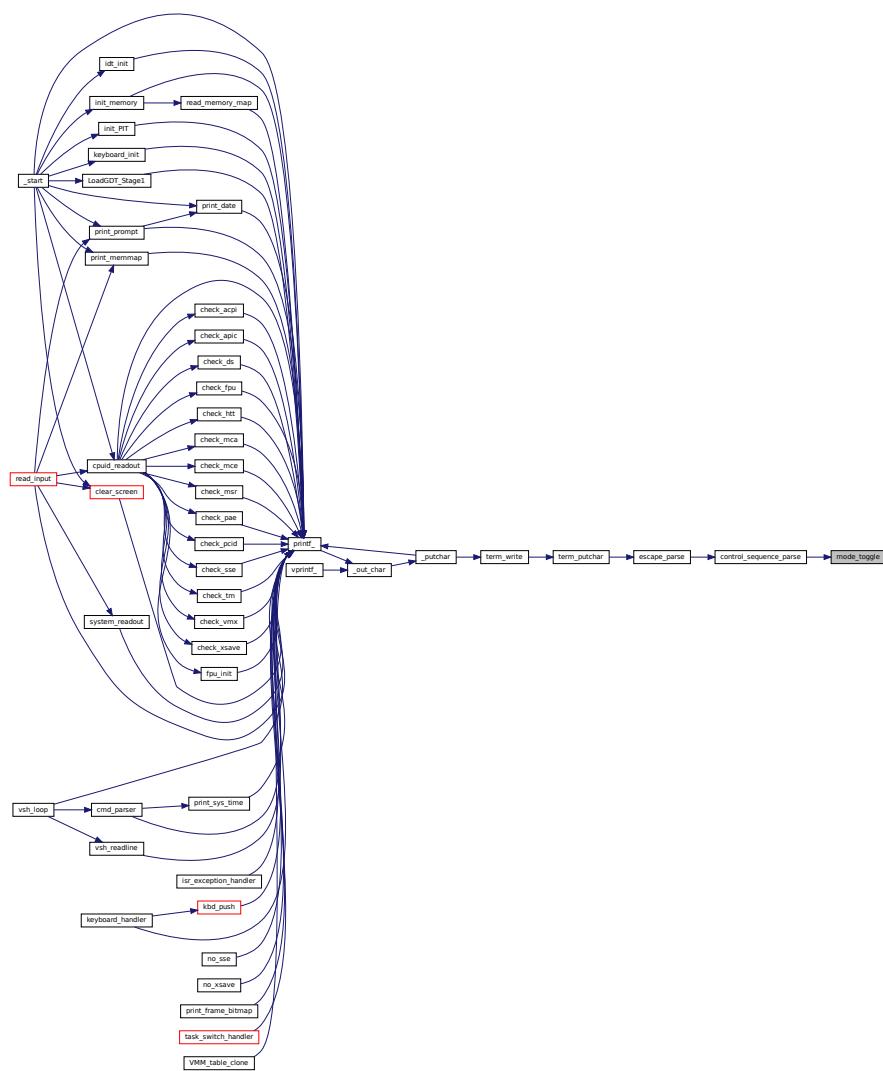


#### 4.137.2.6 mode\_toggle()

```
static void mode_toggle (
    struct term_context * ctx,
    uint8_t c ) [static]
```

Definition at line 402 of file [term.c](#).

Here is the caller graph for this function:

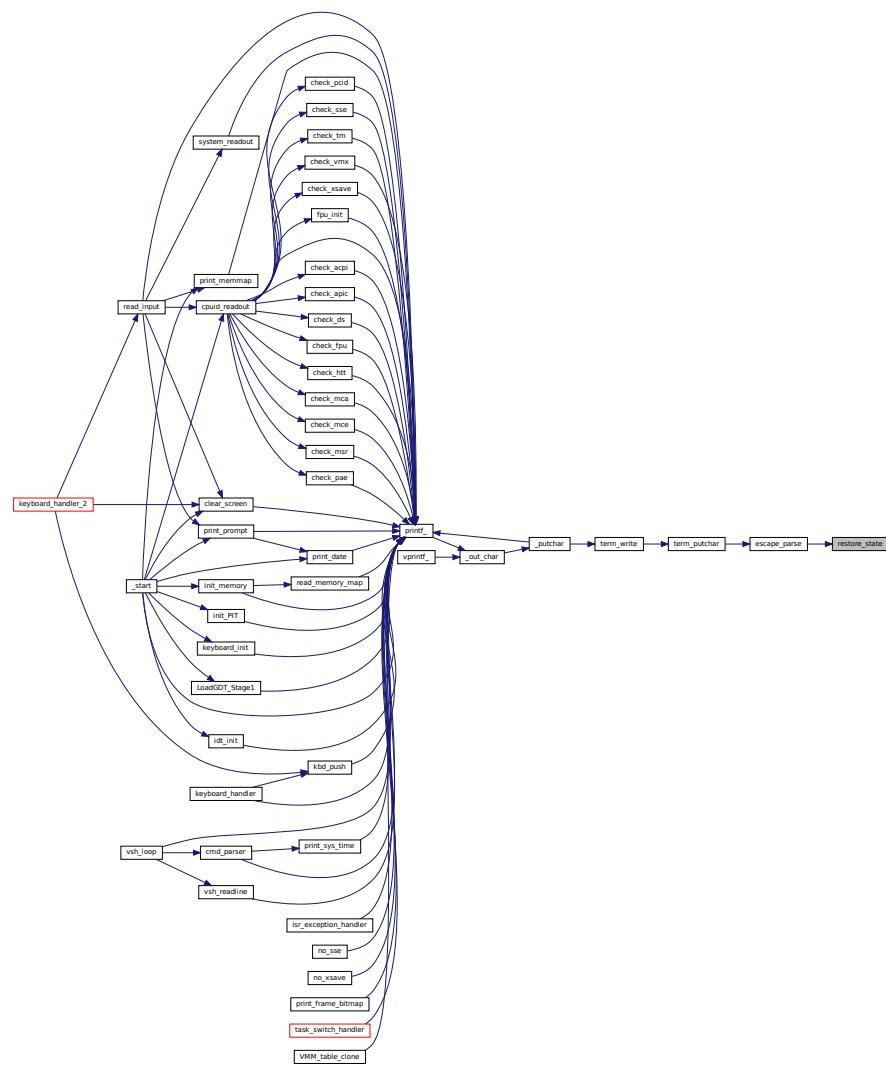


#### 4.137.2.7 restore\_state()

```
static void restore_state (
    struct term_context * ctx ) [static]
```

Definition at line 753 of file [term.c](#).

Here is the caller graph for this function:



#### 4.137.2.8 save\_state()

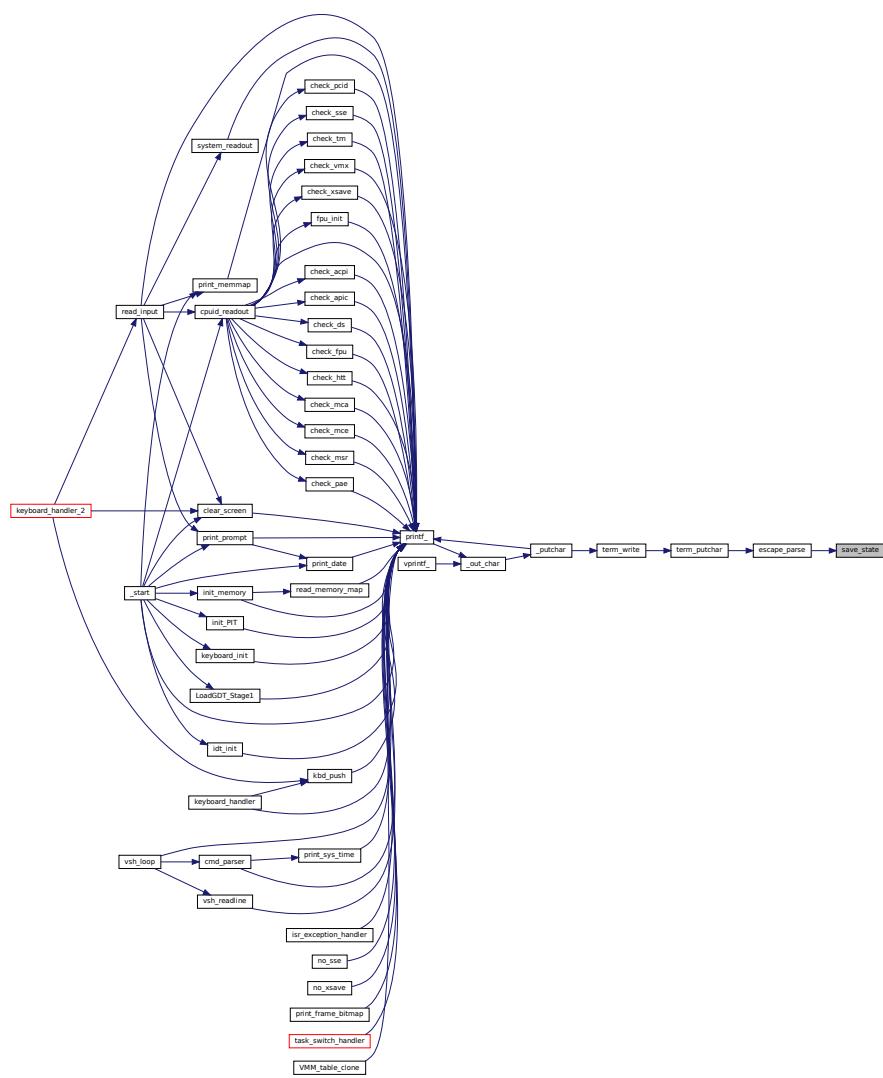
```

static void save_state (
    struct term_context * ctx ) [static]

```

Definition at line 763 of file [term.c](#).

Here is the caller graph for this function:

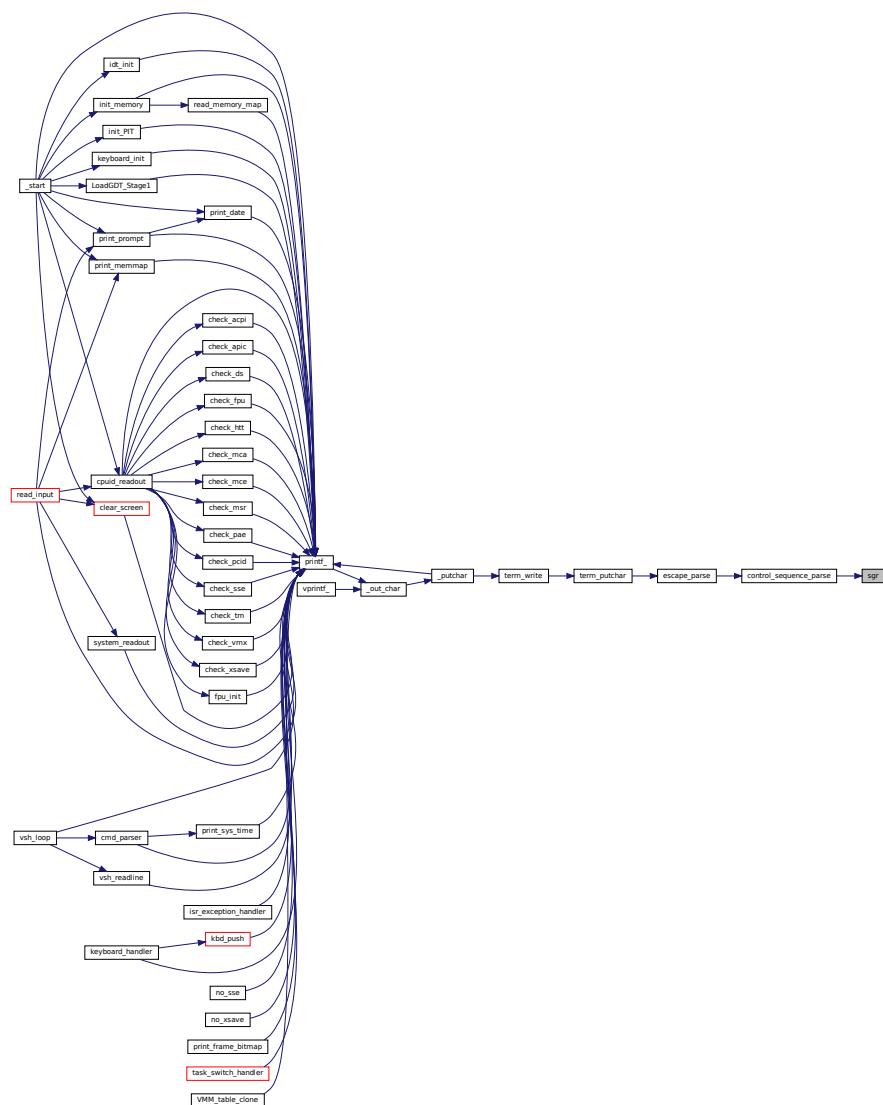


#### 4.137.2.9 sgr()

```
static void sgr (
    struct term_context * ctx ) [static]
```

Definition at line 90 of file [term.c](#).

Here is the caller graph for this function:

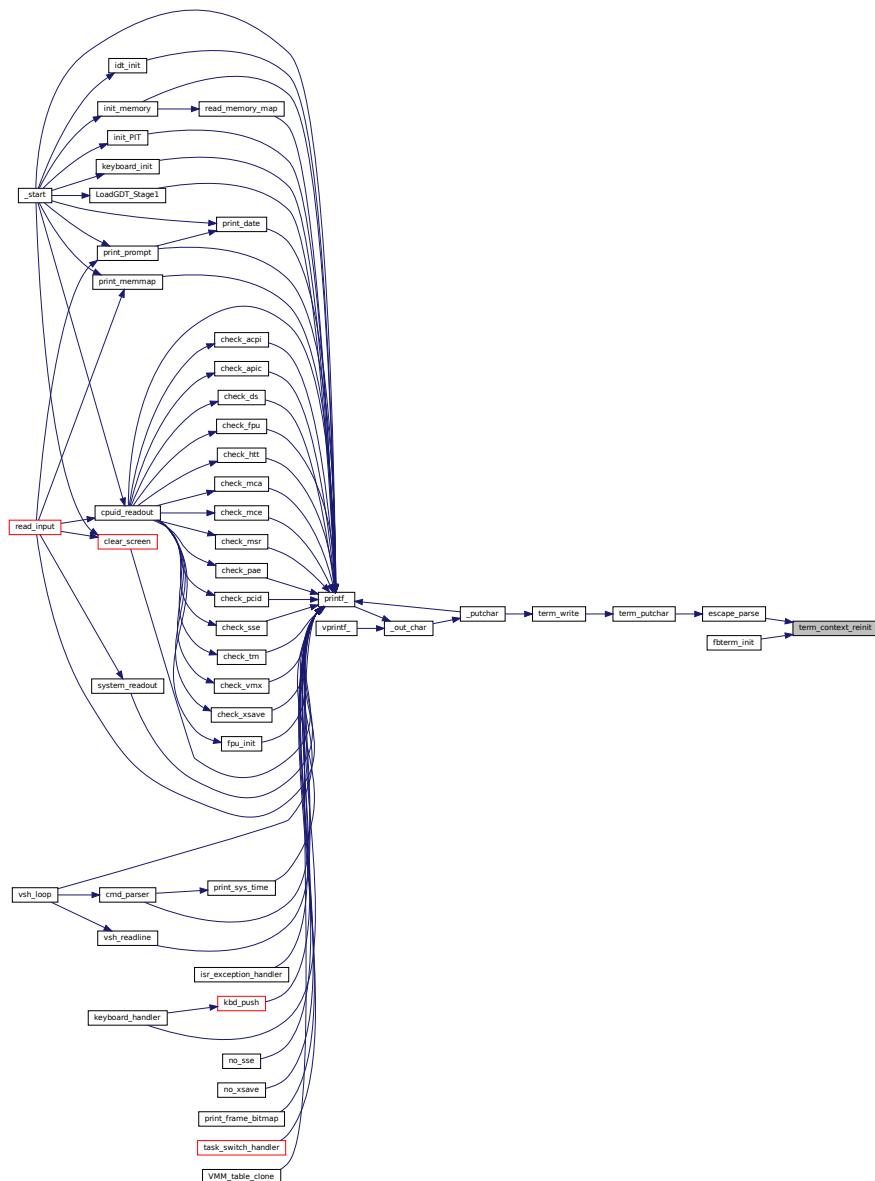


#### 4.137.2.10 `term_context_reinit()`

```
void term_context_reinit (
    struct term_context * ctx )
```

Definition at line 46 of file [term.c](#).

Here is the caller graph for this function:

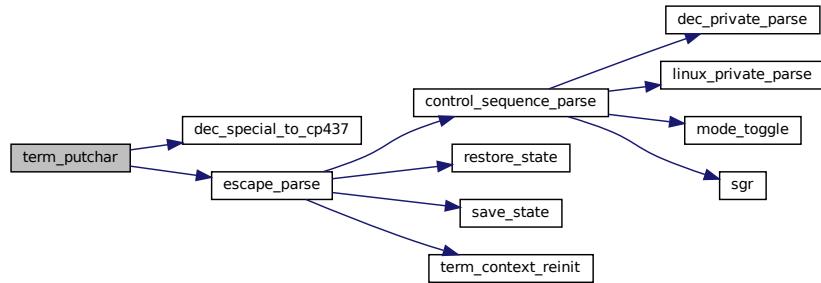


#### 4.137.2.11 `term_putchar()`

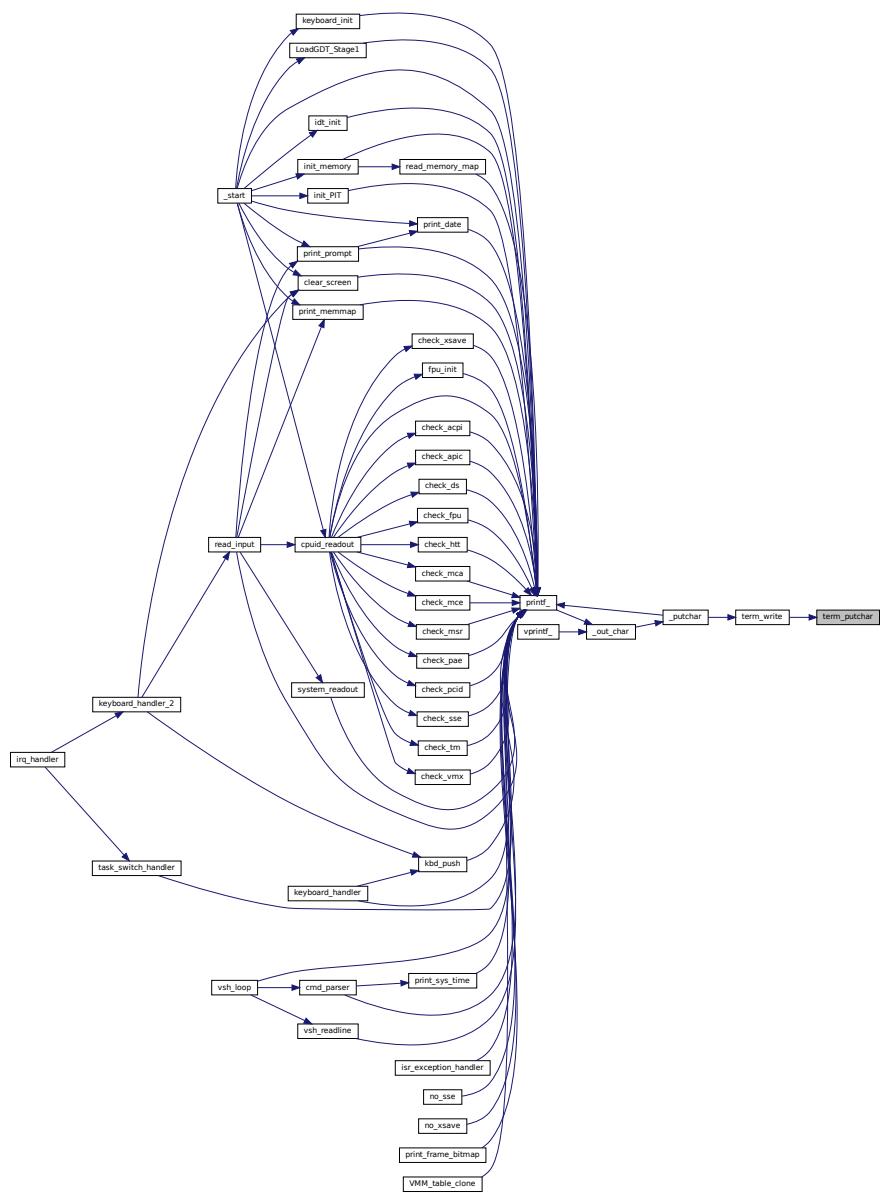
```
static void term_putchar (
    struct term_context * ctx,
    uint8_t c ) [static]
```

Definition at line 932 of file [term.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

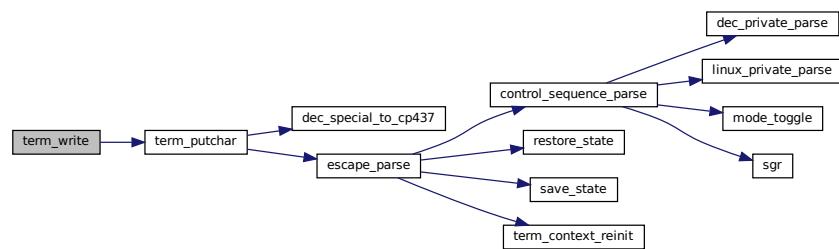


#### 4.137.2.12 term\_write()

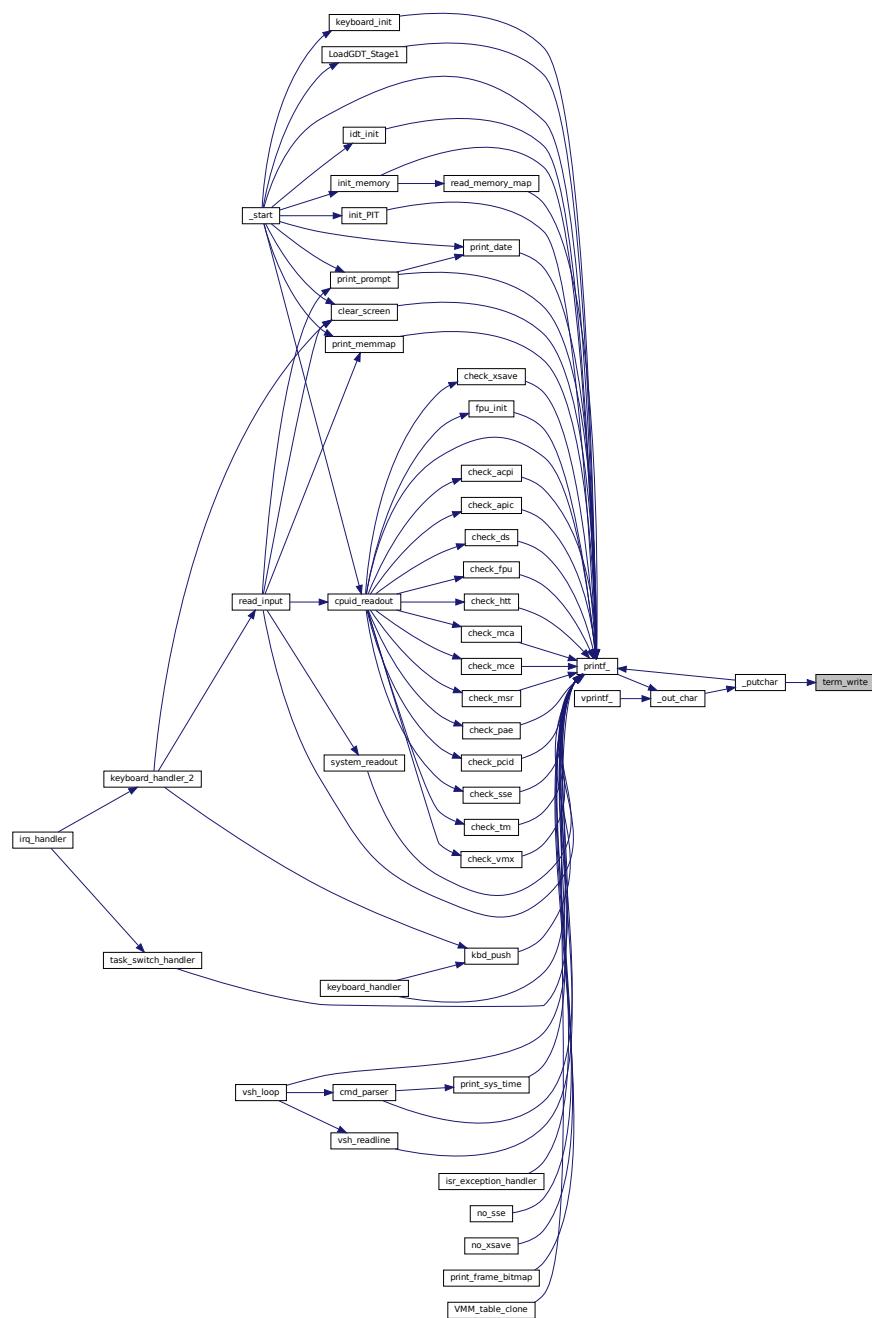
```
void term_write (
    struct term_context * ctx,
    const char * buf,
    size_t count )
```

Definition at line 75 of file [term.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 4.137.3 Variable Documentation

#### 4.137.3.1 col256

```
const uint32_t col256[ ] [static]
```

Definition at line 8 of file [term.c](#).

## 4.138 term.c

```

00001 #include <stdint.h>
00002 #include <stddef.h>
00003 #include <stdbool.h>
00004
00005 #include "../include/terminal/term.h"
00006 #include "../include/serial.h"
00007
00008 static const uint32_t col256[] = {
00009     0x000000, 0x00005f, 0x000087, 0x0000af, 0x0000d7, 0x0000ff, 0x005f00, 0x005f5f,
00010     0x005f87, 0x005faf, 0x005fd7, 0x005fff, 0x008700, 0x00875f, 0x008787, 0x0087af,
00011     0x0087d7, 0x0087ff, 0x00af00, 0x00af5f, 0x00af87, 0x00afaf, 0x00afd7, 0x00afff,
00012     0x00d700, 0x00d75f, 0x00d787, 0x00d7af, 0x00d7d7, 0x00d7ff, 0x00ff00, 0x00ff5f,
00013     0x00ff87, 0x00ffaf, 0x00ffd7, 0x00ffff, 0x5f0000, 0x5f005f, 0x5f0087, 0x5f00af,
00014     0x5f00d7, 0x5f00ff, 0x5f5f00, 0x5f5f5f, 0x5f5f87, 0x5f5faf, 0x5f5fd7, 0x5f5fff,
00015     0x5f8700, 0x5f875f, 0x5f8787, 0x5f87af, 0x5f87d7, 0x5f87ff, 0x5faf00, 0x5faf5f,
00016     0x5faf87, 0x5fafaf, 0x5fafd7, 0x5fafff, 0x5fd700, 0x5fd75f, 0x5fd787, 0x5fd7af,
00017     0x5fd7d7, 0x5fd7ff, 0x5fff00, 0x5fff5f, 0x5fff87, 0x5fffff, 0x5fffff, 0x5fffff,
00018     0x870000, 0x87005f, 0x870087, 0x8700af, 0x8700d7, 0x8700ff, 0x875f00, 0x875f5f,
00019     0x875f87, 0x875faf, 0x875fd7, 0x875fff, 0x878700, 0x87875f, 0x878787, 0x8787af,
00020     0x8787d7, 0x8787ff, 0x87af00, 0x87af5f, 0x87af87, 0x87afaf, 0x87afd7, 0x87afff,
00021     0x87d700, 0x87d75f, 0x87d787, 0x87d7af, 0x87d7d7, 0x87d7ff, 0x87ff00, 0x87ff5f,
00022     0x87ff87, 0x87ffaf, 0x87ffd7, 0x87ffff, 0xaf0000, 0xaf005f, 0xaf0087, 0xaf00af,
00023     0xaf00d7, 0xaf00ff, 0xaf5f00, 0xaf5f5f, 0xaf5f87, 0xaf5faf, 0xaf5fd7, 0xaf5fff,
00024     0xaf8700, 0xaf875f, 0xaf8787, 0xaf87af, 0xaf87d7, 0xaf87ff, 0xafaf00, 0xafaf5f,
00025     0xafaf87, 0xafafaf, 0xafafd7, 0xafafff, 0xafd700, 0xafd75f, 0xafd787, 0xafd7af,
00026     0xafd7d7, 0xafd7ff, 0xafff00, 0xafff5f, 0xafff87, 0xafffaf, 0xafffd7, 0xafffff,
00027     0xd70000, 0xd7005f, 0xd70087, 0xd700af, 0xd700d7, 0xd700ff, 0xd75f00, 0xd75f5f,
00028     0xd75f87, 0xd75faf, 0xd75fd7, 0xd75fff, 0xd78700, 0xd7875f, 0xd78787, 0xd787af,
00029     0xd787d7, 0xd787ff, 0xd7af00, 0xd7af5f, 0xd7af87, 0xd7afaf, 0xd7afd7, 0xd7afff,
00030     0xd7d700, 0xd7d75f, 0xd7d787, 0xd7d7af, 0xd7d7d7, 0xd7d7ff, 0xd7ff00, 0xd7ff5f,
00031     0xd7ff87, 0xd7ffaf, 0xd7ffd7, 0xd7ffff, 0xff0000, 0xff005f, 0xff0087, 0xff00af,
00032     0xff00d7, 0xff00ff, 0xff5f00, 0xff5f5f, 0xff5f87, 0xff5faf, 0xff5fd7, 0xff5fff,
00033     0xff8700, 0xff875f, 0xff8787, 0xff87af, 0xff87d7, 0xff87ff, 0xffaf00, 0xffaf5f,
00034     0xfffd87, 0xfffafaf, 0xfffad7, 0xffafff, 0xfffd700, 0xfffd75f, 0xfffd787, 0xfffd7af,
00035     0xfffd7d7, 0xfffd7ff, 0xffff00, 0xffff5f, 0xffff87, 0xffffaf, 0xfffffd7, 0xfffffff,
00036     0x808080, 0x121212, 0x1c1c1c, 0x262626, 0x303030, 0x3a3a3a, 0x444444, 0x4e4e4e,
00037     0x585858, 0x626262, 0x6c6c6c, 0x767676, 0x808080, 0x8a8a8a, 0x949494, 0x9e9e9e,
00038     0xa8a8a8, 0xb2b2b2, 0xbcbbc, 0xc6c6c6, 0xd0d0d0, 0xdadada, 0xe4e4e4, 0xeeeeee};
00039
00040 // Tries to implement this standard for terminfo
00041 // https://man7.org/linux/man-pages/man4/console_codes.4.html
00042
00043 #define CHARSET_DEFAULT 0
00044 #define CHARSET_DEC_SPECIAL 1
00045
00046 void term_context_reinit(struct term_context *ctx)
00047 {
00048     ctx->tab_size = 8;
00049     ctx->autoflush = true;
00050     ctx->scroll_enabled = true;
00051     ctx->control_sequence = false;
00052     ctx->csi = false;
00053     ctx->escape = false;
00054     ctx->rrr = false;
00055     ctx->discard_next = false;
00056     ctx->bold = false;
00057     ctx->reverse_video = false;
00058     ctx->dec_private = false;
00059     ctx->insert_mode = false;
00060     ctx->g_select = 0;
00061     ctx->charsets[0] = CHARSET_DEFAULT;
00062     ctx->charsets[1] = CHARSET_DEC_SPECIAL;
00063     ctx->current_charset = 0;
00064     ctx->escape_offset = 0;
00065     ctx->esc_values_i = 0;
00066     ctx->saved_cursor_x = 0;
00067     ctx->saved_cursor_y = 0;
00068     ctx->current_primary = (size_t)-1;
00069     ctx->scroll_top_margin = 0;
00070     ctx->scroll_bottom_margin = ctx->rows;
00071 }
00072
00073 static void term_putchar(struct term_context *ctx, uint8_t c);
00074
00075 void term_write(struct term_context *ctx, const char *buf, size_t count)
00076 {
00077
00078     for (size_t i = 0; i < count; i++)
00079     {
00080         term_putchar(ctx, buf[i]);
00081     }
00082
00083     if (ctx->autoflush)
00084     {

```

```
00086         ctx->double_buffer_flush(ctx);
00087     }
00088 }
00089
00090 static void sgr(struct term_context *ctx)
00091 {
00092     size_t i = 0;
00093
00094     if (!ctx->esc_values_i)
00095         goto def;
00096
00097     for (; i < ctx->esc_values_i; i++)
00098     {
00099         size_t offset;
00100
00101         if (ctx->esc_values[i] == 0)
00102         {
00103             def:
00104                 if (ctx->reverse_video)
00105                 {
00106                     ctx->reverse_video = false;
00107                     ctx->swap_palette(ctx);
00108                 }
00109                 ctx->bold = false;
00110                 ctx->current_primary = (size_t)-1;
00111                 ctx->set_text_bg_default(ctx);
00112                 ctx->set_text_fg_default(ctx);
00113                 continue;
00114         }
00115
00116         else if (ctx->esc_values[i] == 1)
00117         {
00118             ctx->bold = true;
00119             if (ctx->current_primary != (size_t)-1)
00120             {
00121                 if (!ctx->reverse_video)
00122                 {
00123                     ctx->set_text_fg_bright(ctx, ctx->current_primary);
00124                 }
00125                 else
00126                 {
00127                     ctx->set_text_bg_bright(ctx, ctx->current_primary);
00128                 }
00129             }
00130             continue;
00131         }
00132
00133         else if (ctx->esc_values[i] == 22)
00134         {
00135             ctx->bold = false;
00136             if (ctx->current_primary != (size_t)-1)
00137             {
00138                 if (!ctx->reverse_video)
00139                 {
00140                     ctx->set_text_fg(ctx, ctx->current_primary);
00141                 }
00142                 else
00143                 {
00144                     ctx->set_text_bg(ctx, ctx->current_primary);
00145                 }
00146             }
00147             continue;
00148         }
00149
00150         else if (ctx->esc_values[i] >= 30 && ctx->esc_values[i] <= 37)
00151         {
00152             offset = 30;
00153             ctx->current_primary = ctx->esc_values[i] - offset;
00154
00155             if (ctx->reverse_video)
00156             {
00157                 goto set_bg;
00158             }
00159
00160             set_fg:
00161             if (ctx->bold && !ctx->reverse_video)
00162             {
00163                 ctx->set_text_fg_bright(ctx, ctx->esc_values[i] - offset);
00164             }
00165             else
00166             {
00167                 ctx->set_text_fg(ctx, ctx->esc_values[i] - offset);
00168             }
00169             continue;
00170         }
00171     }
00172     else if (ctx->esc_values[i] >= 40 && ctx->esc_values[i] <= 47)
```

```
00173     {
00174         offset = 40;
00175         if (ctx->reverse_video)
00176         {
00177             goto set_fg;
00178         }
00179
00180         set_bg:
00181             if (ctx->bold && ctx->reverse_video)
00182             {
00183                 ctx->set_text_bg_bright(ctx, ctx->esc_values[i] - offset);
00184             }
00185             else
00186             {
00187                 ctx->set_text_bg(ctx, ctx->esc_values[i] - offset);
00188             }
00189             continue;
00190     }
00191
00192     else if (ctx->esc_values[i] >= 90 && ctx->esc_values[i] <= 97)
00193     {
00194         offset = 90;
00195         ctx->current_primary = ctx->esc_values[i] - offset;
00196
00197         if (ctx->reverse_video)
00198         {
00199             goto set_bg_bright;
00200         }
00201
00202         set_fg_bright:
00203             ctx->set_text_fg_bright(ctx, ctx->esc_values[i] - offset);
00204             continue;
00205     }
00206
00207     else if (ctx->esc_values[i] >= 100 && ctx->esc_values[i] <= 107)
00208     {
00209         offset = 100;
00210         if (ctx->reverse_video)
00211         {
00212             goto set_fg_bright;
00213         }
00214
00215         set_bg_bright:
00216             ctx->set_text_bg_bright(ctx, ctx->esc_values[i] - offset);
00217             continue;
00218     }
00219
00220     else if (ctx->esc_values[i] == 39)
00221     {
00222         ctx->current_primary = (size_t)-1;
00223
00224         if (ctx->reverse_video)
00225         {
00226             ctx->swap_palette(ctx);
00227         }
00228
00229         ctx->set_text_fg_default(ctx);
00230
00231         if (ctx->reverse_video)
00232         {
00233             ctx->swap_palette(ctx);
00234         }
00235
00236         continue;
00237     }
00238
00239     else if (ctx->esc_values[i] == 49)
00240     {
00241         if (ctx->reverse_video)
00242         {
00243             ctx->swap_palette(ctx);
00244         }
00245
00246         ctx->set_text_bg_default(ctx);
00247
00248         if (ctx->reverse_video)
00249         {
00250             ctx->swap_palette(ctx);
00251         }
00252
00253         continue;
00254     }
00255
00256     else if (ctx->esc_values[i] == 7)
00257     {
00258         if (!ctx->reverse_video)
00259         {
```

```

00260         ctx->reverse_video = true;
00261         ctx->swap_palette(ctx);
00262     }
00263     continue;
00264 }
00265
00266 else if (ctx->esc_values[i] == 27)
00267 {
00268     if (ctx->reverse_video)
00269     {
00270         ctx->reverse_video = false;
00271         ctx->swap_palette(ctx);
00272     }
00273     continue;
00274 }
00275
00276 // 256/RGB
00277 else if (ctx->esc_values[i] == 38 || ctx->esc_values[i] == 48)
00278 {
00279     bool fg = ctx->esc_values[i] == 38;
00280
00281     i++;
00282     if (i >= ctx->esc_values_i)
00283     {
00284         break;
00285     }
00286
00287     switch (ctx->esc_values[i])
00288     {
00289     case 2:
00290     { // RGB
00291         if (i + 3 >= ctx->esc_values_i)
00292         {
00293             goto out;
00294         }
00295
00296         uint32_t rgb_value = 0;
00297
00298         rgb_value |= ctx->esc_values[i + 1] << 16;
00299         rgb_value |= ctx->esc_values[i + 2] << 8;
00300         rgb_value |= ctx->esc_values[i + 3];
00301
00302         i += 3;
00303
00304         (fg ? ctx->set_text_fg_rgb : ctx->set_text_bg_rgb)(ctx, rgb_value);
00305
00306         break;
00307     }
00308     case 5:
00309     { // 256 colors
00310         if (i + 1 >= ctx->esc_values_i)
00311         {
00312             goto out;
00313         }
00314
00315         uint32_t col = ctx->esc_values[i + 1];
00316
00317         i++;
00318
00319         if (col < 8)
00320         {
00321             (fg ? ctx->set_text_fg : ctx->set_text_bg)(ctx, col);
00322         }
00323         else if (col < 16)
00324         {
00325             (fg ? ctx->set_text_fg_bright : ctx->set_text_bg_bright)(ctx, col - 8);
00326         }
00327         else
00328         {
00329             uint32_t rgb_value = col256[col - 16];
00330             (fg ? ctx->set_text_fg_rgb : ctx->set_text_bg_rgb)(ctx, rgb_value);
00331         }
00332
00333         break;
00334     }
00335     default:
00336         continue;
00337     }
00338 }
00339 }
00340
00341 out:;
00342 }
00343
00344 static void dec_private_parse(struct term_context *ctx, uint8_t c)
00345 {
00346     ctx->dec_private = false;

```

```
00347
00348     if (ctx->esc_values_i == 0)
00349     {
00350         return;
00351     }
00352
00353     bool set;
00354
00355     switch (c)
00356     {
00357         case 'h':
00358             set = true;
00359             break;
00360         case 'l':
00361             set = false;
00362             break;
00363         default:
00364             return;
00365     }
00366
00367     switch (ctx->esc_values[0])
00368     {
00369         case 25:
00370         {
00371             if (set)
00372             {
00373                 ctx->enable_cursor(ctx);
00374             }
00375             else
00376             {
00377                 ctx->disable_cursor(ctx);
00378             }
00379             return;
00380         }
00381     }
00382
00383     if (ctx->callback != NULL)
00384     {
00385         ctx->callback(ctx, TERM_CB_DEC, ctx->esc_values_i, (uintptr_t)ctx->esc_values, c);
00386     }
00387 }
00388
00389 static void linux_private_parse(struct term_context *ctx)
00390 {
00391     if (ctx->esc_values_i == 0)
00392     {
00393         return;
00394     }
00395
00396     if (ctx->callback != NULL)
00397     {
00398         ctx->callback(ctx, TERM_CB_LINUX, ctx->esc_values_i, (uintptr_t)ctx->esc_values, 0);
00399     }
00400 }
00401
00402 static void mode_toggle(struct term_context *ctx, uint8_t c)
00403 {
00404     if (ctx->esc_values_i == 0)
00405     {
00406         return;
00407     }
00408
00409     bool set;
00410
00411     switch (c)
00412     {
00413         case 'h':
00414             set = true;
00415             break;
00416         case 'l':
00417             set = false;
00418             break;
00419         default:
00420             return;
00421     }
00422
00423     switch (ctx->esc_values[0])
00424     {
00425         case 4:
00426             ctx->insert_mode = set;
00427             return;
00428     }
00429
00430     if (ctx->callback != NULL)
00431     {
00432         ctx->callback(ctx, TERM_CB_MODE, ctx->esc_values_i, (uintptr_t)ctx->esc_values, c);
00433     }
```

```

00434 }
00435
00436 static void control_sequence_parse(struct term_context *ctx, uint8_t c)
00437 {
00438     if (ctx->escape_offset == 2)
00439     {
00440         switch (c)
00441         {
00442             case '[':
00443                 ctx->discard_next = true;
00444                 goto cleanup;
00445             case '?':
00446                 ctx->dec_private = true;
00447                 return;
00448         }
00449     }
00450
00451     if (c >= '0' && c <= '9')
00452     {
00453         if (ctx->esc_values_i == TERM_MAX_ESC_VALUES)
00454         {
00455             return;
00456         }
00457         ctx->rrr = true;
00458         ctx->esc_values[ctx->esc_values_i] *= 10;
00459         ctx->esc_values[ctx->esc_values_i] += c - '0';
00460         return;
00461     }
00462
00463     if (ctx->rrr == true)
00464     {
00465         ctx->esc_values_i++;
00466         ctx->rrr = false;
00467         if (c == ';')
00468             return;
00469     }
00470     else if (c == ',')
00471     {
00472         if (ctx->esc_values_i == TERM_MAX_ESC_VALUES)
00473         {
00474             return;
00475         }
00476         ctx->esc_values[ctx->esc_values_i] = 0;
00477         ctx->esc_values_i++;
00478         return;
00479     }
00480
00481     size_t esc_default;
00482     switch (c)
00483     {
00484         case 'J':
00485         case 'K':
00486         case 'q':
00487             esc_default = 0;
00488             break;
00489         default:
00490             esc_default = 1;
00491             break;
00492     }
00493
00494     for (size_t i = ctx->esc_values_i; i < TERM_MAX_ESC_VALUES; i++)
00495     {
00496         ctx->esc_values[i] = esc_default;
00497     }
00498
00499     if (ctx->dec_private == true)
00500     {
00501         dec_private_parse(ctx, c);
00502         goto cleanup;
00503     }
00504
00505     bool r = ctx->scroll_enabled;
00506     ctx->scroll_enabled = false;
00507     size_t x, y;
00508     ctx->get_cursor_pos(ctx, &x, &y);
00509
00510     switch (c)
00511     {
00512         case 'F':
00513             x = 0;
00514             // FALLTHRU
00515         case 'A':
00516         {
00517             if (ctx->esc_values[0] > y)
00518                 ctx->esc_values[0] = y;
00519             size_t orig_y = y;
00520             size_t dest_y = y - ctx->esc_values[0];

```

```
00521     bool will_be_in_scroll_region = false;
00522     if ((ctx->scroll_top_margin >= dest_y && ctx->scroll_top_margin <= orig_y) ||
00523         (ctx->scroll_bottom_margin >= dest_y && ctx->scroll_bottom_margin <= orig_y))
00524     {
00525         will_be_in_scroll_region = true;
00526     }
00527     if (will_be_in_scroll_region && dest_y < ctx->scroll_top_margin)
00528     {
00529         dest_y = ctx->scroll_top_margin;
00530     }
00531     ctx->set_cursor_pos(ctx, x, dest_y);
00532     break;
00533 case 'E':
00534     x = 0;
00535     // FALLTHRU
00536 case 'e':
00537 case 'B':
00538 {
00539     if (y + ctx->esc_values[0] > ctx->rows - 1)
00540         ctx->esc_values[0] = (ctx->rows - 1) - y;
00541     size_t orig_y = y;
00542     size_t dest_y = y + ctx->esc_values[0];
00543     bool will_be_in_scroll_region = false;
00544     if ((ctx->scroll_top_margin >= orig_y && ctx->scroll_top_margin <= dest_y) ||
00545         (ctx->scroll_bottom_margin >= orig_y && ctx->scroll_bottom_margin <= dest_y))
00546     {
00547         will_be_in_scroll_region = true;
00548     }
00549     if (will_be_in_scroll_region && dest_y >= ctx->scroll_bottom_margin)
00550     {
00551         dest_y = ctx->scroll_bottom_margin - 1;
00552     }
00553     ctx->set_cursor_pos(ctx, x, dest_y);
00554     break;
00555 case 'a':
00556 case 'C':
00557     if (x + ctx->esc_values[0] > ctx->cols - 1)
00558         ctx->esc_values[0] = (ctx->cols - 1) - x;
00559     ctx->set_cursor_pos(ctx, x + ctx->esc_values[0], y);
00560     break;
00561 case 'D':
00562     if (ctx->esc_values[0] > x)
00563         ctx->esc_values[0] = x;
00564     ctx->set_cursor_pos(ctx, x - ctx->esc_values[0], y);
00565     break;
00566 case 'c':
00567     if (ctx->callback != NULL)
00568     {
00569         ctx->callback(ctx, TERM_CB_PRIVATE_ID, 0, 0, 0);
00570     }
00571     break;
00572 case 'd':
00573     ctx->esc_values[0] -= 1;
00574     if (ctx->esc_values[0] >= ctx->rows)
00575         ctx->esc_values[0] = ctx->rows - 1;
00576     ctx->set_cursor_pos(ctx, x, ctx->esc_values[0]);
00577     break;
00578 case 'G':
00579 case '`':
00580     ctx->esc_values[0] -= 1;
00581     if (ctx->esc_values[0] >= ctx->cols)
00582         ctx->esc_values[0] = ctx->cols - 1;
00583     ctx->set_cursor_pos(ctx, ctx->esc_values[0], y);
00584     break;
00585 case 'H':
00586 case 'f':
00587     ctx->esc_values[0] -= 1;
00588     ctx->esc_values[1] -= 1;
00589     if (ctx->esc_values[1] >= ctx->cols)
00590         ctx->esc_values[1] = ctx->cols - 1;
00591     if (ctx->esc_values[0] >= ctx->rows)
00592         ctx->esc_values[0] = ctx->rows - 1;
00593     ctx->set_cursor_pos(ctx, ctx->esc_values[1], ctx->esc_values[0]);
00594     break;
00595 case 'n':
00596     switch (ctx->esc_values[0])
00597     {
00598         case 5:
00599             if (ctx->callback != NULL)
00600             {
00601                 ctx->callback(ctx, TERM_CB_STATUS_REPORT, 0, 0, 0);
00602             }
00603             break;
00604         case 6:
00605             if (ctx->callback != NULL)
```

```

00606      {
00607          ctx->callback(ctx, TERM_CB_POS_REPORT, x + 1, y + 1, 0);
00608      }
00609      break;
00610  }
00611  break;
00612 case 'q':
00613     if (ctx->callback != NULL)
00614     {
00615         ctx->callback(ctx, TERM_CB_KBD_LEDS, ctx->esc_values[0], 0, 0);
00616     }
00617     break;
00618 case 'J':
00619     switch (ctx->esc_values[0])
00620     {
00621     case 0:
00622     {
00623         size_t rows_remaining = ctx->rows - (y + 1);
00624         size_t cols_diff = ctx->cols - (x + 1);
00625         size_t to_clear = rows_remaining * ctx->cols + cols_diff;
00626         for (size_t i = 0; i < to_clear; i++)
00627         {
00628             ctx->raw_putchar(ctx, ' ');
00629         }
00630         ctx->set_cursor_pos(ctx, x, y);
00631         break;
00632     }
00633     case 1:
00634     {
00635         ctx->set_cursor_pos(ctx, 0, 0);
00636         bool b = false;
00637         for (size_t yc = 0; yc < ctx->rows; yc++)
00638         {
00639             for (size_t xc = 0; xc < ctx->cols; xc++)
00640             {
00641                 ctx->raw_putchar(ctx, ' ');
00642                 if (xc == x && yc == y)
00643                 {
00644                     ctx->set_cursor_pos(ctx, x, y);
00645                     b = true;
00646                     break;
00647                 }
00648             }
00649             if (b == true)
00650                 break;
00651         }
00652         break;
00653     }
00654     case 2:
00655     case 3:
00656         ctx->clear(ctx, false);
00657         break;
00658     }
00659     break;
00660 case '@':
00661     for (size_t i = ctx->cols - 1;; i--)
00662     {
00663         ctx->move_character(ctx, i + ctx->esc_values[0], y, i, y);
00664         ctx->set_cursor_pos(ctx, i, y);
00665         ctx->raw_putchar(ctx, ' ');
00666         if (i == x)
00667         {
00668             break;
00669         }
00670     }
00671     ctx->set_cursor_pos(ctx, x, y);
00672     break;
00673 case 'P':
00674     for (size_t i = x + ctx->esc_values[0]; i < ctx->cols; i++)
00675         ctx->move_character(ctx, i - ctx->esc_values[0], y, i, y);
00676     ctx->set_cursor_pos(ctx, ctx->cols - ctx->esc_values[0], y);
00677 // FALLTHRU
00678 case 'X':
00679     for (size_t i = 0; i < ctx->esc_values[0]; i++)
00680         ctx->raw_putchar(ctx, ' ');
00681     ctx->set_cursor_pos(ctx, x, y);
00682     break;
00683 case 'm':
00684     sgr(ctx);
00685     break;
00686 case 's':
00687     ctx->get_cursor_pos(ctx, &ctx->saved_cursor_x, &ctx->saved_cursor_y);
00688     break;
00689 case 'u':
00690     ctx->set_cursor_pos(ctx, ctx->saved_cursor_x, ctx->saved_cursor_y);
00691     break;
00692 case 'K':

```

```
00693     switch (ctx->esc_values[0])
00694     {
00695         case 0:
00696         {
00697             for (size_t i = x; i < ctx->cols; i++)
00698                 ctx->raw_putchar(ctx, ' ');
00699             ctx->set_cursor_pos(ctx, x, y);
00700             break;
00701         }
00702         case 1:
00703         {
00704             ctx->set_cursor_pos(ctx, 0, y);
00705             for (size_t i = 0; i < x; i++)
00706                 ctx->raw_putchar(ctx, ' ');
00707             break;
00708         }
00709         case 2:
00710         {
00711             ctx->set_cursor_pos(ctx, 0, y);
00712             for (size_t i = 0; i < ctx->cols; i++)
00713                 ctx->raw_putchar(ctx, ' ');
00714             ctx->set_cursor_pos(ctx, x, y);
00715             break;
00716         }
00717     }
00718     break;
00719     case 'r':
00720     {
00721         ctx->scroll_top_margin = 0;
00722         ctx->scroll_bottom_margin = ctx->rows;
00723         if (ctx->esc_values_i > 0)
00724         {
00725             ctx->scroll_top_margin = ctx->esc_values[0] - 1;
00726         }
00727         if (ctx->esc_values_i > 1)
00728         {
00729             ctx->scroll_bottom_margin = ctx->esc_values[1];
00730         }
00731         if (ctx->scroll_top_margin >= ctx->rows || ctx->scroll_bottom_margin > ctx->rows ||
00732             ctx->scroll_top_margin >= (ctx->scroll_bottom_margin - 1))
00733         {
00734             ctx->scroll_top_margin = 0;
00735             ctx->scroll_bottom_margin = ctx->rows;
00736         }
00737         ctx->set_cursor_pos(ctx, 0, 0);
00738         break;
00739     case 'l':
00740     case 'h':
00741         mode_toggle(ctx, c);
00742         break;
00743     case ']':
00744         linux_private_parse(ctx);
00745         break;
00746     }
00747     ctx->scroll_enabled = r;
00748 cleanup:
00749     ctx->control_sequence = false;
00750     ctx->escape = false;
00751 }
00752
00753 static void restore_state(struct term_context *ctx)
00754 {
00755     ctx->bold = ctx->saved_state_bold;
00756     ctx->reverse_video = ctx->saved_state_reverse_video;
00757     ctx->current_charset = ctx->saved_state_current_charset;
00758     ctx->current_primary = ctx->saved_state_current_primary;
00759
00760     ctx->restore_state(ctx);
00761 }
00762
00763 static void save_state(struct term_context *ctx)
00764 {
00765     ctx->save_state(ctx);
00766
00767     ctx->saved_state_bold = ctx->bold;
00768     ctx->saved_state_reverse_video = ctx->reverse_video;
00769     ctx->saved_state_current_charset = ctx->current_charset;
00770     ctx->saved_state_current_primary = ctx->current_primary;
00771 }
00772
00773 static void escape_parse(struct term_context *ctx, uint8_t c)
00774 {
00775     ctx->escape_offset++;
00776
00777     if (ctx->control_sequence == true)
00778     {
```

```
00779     control_sequence_parse(ctx, c);
00780     return;
00781 }
00782
00783 if (ctx->csi == true)
00784 {
00785     ctx->csi = false;
00786     goto is_csi;
00787 }
00788
00789 size_t x, y;
00790 ctx->get_cursor_pos(ctx, &x, &y);
00791
00792 switch (c)
00793 {
00794 case '[':
00795     is_csi:
00796     for (size_t i = 0; i < TERM_MAX_ESC_VALUES; i++)
00797         ctx->esc_values[i] = 0;
00798     ctx->esc_values_i = 0;
00799     ctx->rrr = false;
00800     ctx->control_sequence = true;
00801     return;
00802 case '7':
00803     save_state(ctx);
00804     break;
00805 case '8':
00806     restore_state(ctx);
00807     break;
00808 case 'c':
00809     term_context_reinit(ctx);
00810     ctx->clear(ctx, true);
00811     break;
00812 case 'D':
00813     if (y == ctx->scroll_bottom_margin - 1)
00814     {
00815         ctx->scroll(ctx);
00816         ctx->set_cursor_pos(ctx, x, y);
00817     }
00818     else
00819     {
00820         ctx->set_cursor_pos(ctx, x, y + 1);
00821     }
00822     break;
00823 case 'E':
00824     if (y == ctx->scroll_bottom_margin - 1)
00825     {
00826         ctx->scroll(ctx);
00827         ctx->set_cursor_pos(ctx, 0, y);
00828     }
00829     else
00830     {
00831         ctx->set_cursor_pos(ctx, 0, y + 1);
00832     }
00833     break;
00834 case 'M':
00835     // "Reverse linefeed"
00836     if (y == ctx->scroll_top_margin)
00837     {
00838         ctx->revscroll(ctx);
00839         ctx->set_cursor_pos(ctx, 0, y);
00840     }
00841     else
00842     {
00843         ctx->set_cursor_pos(ctx, 0, y - 1);
00844     }
00845     break;
00846 case 'Z':
00847     if (ctx->callback != NULL)
00848     {
00849         ctx->callback(ctx, TERM_CB_PRIVATE_ID, 0, 0, 0);
00850     }
00851     break;
00852 case '(':
00853 case ')':
00854     ctx->g_select = c - '\\';
00855     break;
00856 case '\\e':
00857     if (ctx->in_bootloader == true)
00858     {
00859         ctx->raw_putchar(ctx, c);
00860     }
00861     break;
00862 }
00863
00864 ctx->escape = false;
00865 }
```

```
00866
00867 static uint8_t dec_special_to_cp437(uint8_t c)
00868 {
00869     switch (c)
00870     {
00871         case '\`':
00872             return 0x04;
00873         case '0':
00874             return 0xdb;
00875         case '-':
00876             return 0x18;
00877         case ',':
00878             return 0x1b;
00879         case '.':
00880             return 0x19;
00881         case 'a':
00882             return 0xb1;
00883         case 'f':
00884             return 0xf8;
00885         case 'g':
00886             return 0xf1;
00887         case 'h':
00888             return 0xb0;
00889         case 'j':
00890             return 0xd9;
00891         case 'k':
00892             return 0xbf;
00893         case 'l':
00894             return 0xda;
00895         case 'm':
00896             return 0xc0;
00897         case 'n':
00898             return 0xc5;
00899         case 'q':
00900             return 0xc4;
00901         case 's':
00902             return 0x5f;
00903         case 't':
00904             return 0xc3;
00905         case 'u':
00906             return 0xb4;
00907         case 'v':
00908             return 0xc1;
00909         case 'w':
00910             return 0xc2;
00911         case 'x':
00912             return 0xb3;
00913         case 'y':
00914             return 0xf3;
00915         case 'z':
00916             return 0xf2;
00917         case '~':
00918             return 0xfa;
00919         case '_':
00920             return 0xff;
00921         case '+':
00922             return 0x1a;
00923         case '{':
00924             return 0xe3;
00925         case '}':
00926             return 0x9c;
00927     }
00928
00929     return c;
00930 }
00931
00932 static void term_putchar(struct term_context *ctx, uint8_t c)
00933 {
00934     if (ctx->discard_next || (ctx->in_bootloader == false && (c == 0x18 || c == 0x1a)))
00935     {
00936         ctx->discard_next = false;
00937         ctx->escape = false;
00938         ctx->csi = false;
00939         ctx->control_sequence = false;
00940         ctx->g_select = 0;
00941         return;
00942     }
00943
00944     if (ctx->escape == true)
00945     {
00946         escape_parse(ctx, c);
00947         return;
00948     }
00949
00950     if (ctx->g_select)
00951     {
00952         ctx->g_select--;
00953     }
00954 }
```

```

00953     switch (c)
00954     {
00955         case 'B':
00956             ctx->charsets[ctx->g_select] = CHARSET_DEFAULT;
00957             break;
00958         case '0':
00959             ctx->charsets[ctx->g_select] = CHARSET_DEC_SPECIAL;
00960             break;
00961     }
00962     ctx->g_select = 0;
00963     return;
00964 }
00965
00966 size_t x, y;
00967 ctx->get_cursor_pos(ctx, &x, &y);
00968
00969 switch (c)
00970 {
00971     case 0x00:
00972     case 0x7f:
00973         return;
00974     case 0x9b:
00975         ctx->csi = true;
00976         // FALLTHRU
00977     case '\e':
00978         ctx->escape_offset = 0;
00979         ctx->escape = true;
00980         return;
00981     case '\t':
00982         if ((x / ctx->tab_size + 1) >= ctx->cols)
00983         {
00984             ctx->set_cursor_pos(ctx, ctx->cols - 1, y);
00985             return;
00986         }
00987         ctx->set_cursor_pos(ctx, (x / ctx->tab_size + 1) * ctx->tab_size, y);
00988         return;
00989     case 0x0b:
00990     case 0x0c:
00991     case '\n':
00992         if (y == ctx->scroll_bottom_margin - 1)
00993         {
00994             ctx->scroll(ctx);
00995             ctx->set_cursor_pos(ctx, 0, y);
00996         }
00997         else
00998         {
00999             ctx->set_cursor_pos(ctx, 0, y + 1);
01000         }
01001         return;
01002     case '\b':
01003         ctx->set_cursor_pos(ctx, x - 1, y);
01004         return;
01005     case '\r':
01006         ctx->set_cursor_pos(ctx, 0, y);
01007         return;
01008     case '\a':
01009         // The bell is handled by the kernel
01010         if (ctx->callback != NULL)
01011         {
01012             ctx->callback(ctx, TERM_CB_BELL, 0, 0, 0);
01013         }
01014         return;
01015     case 14:
01016         // Move to G1 set
01017         ctx->current_charset = 1;
01018         return;
01019     case 15:
01020         // Move to G0 set
01021         ctx->current_charset = 0;
01022         return;
01023     }
01024
01025     if (ctx->insert_mode == true)
01026     {
01027         for (size_t i = ctx->cols - 1;; i--)
01028         {
01029             ctx->move_character(ctx, i + 1, y, i, y);
01030             if (i == x)
01031             {
01032                 break;
01033             }
01034         }
01035     }
01036
01037     // Translate character set
01038     switch (ctx->charsets[ctx->current_charset])
01039     {

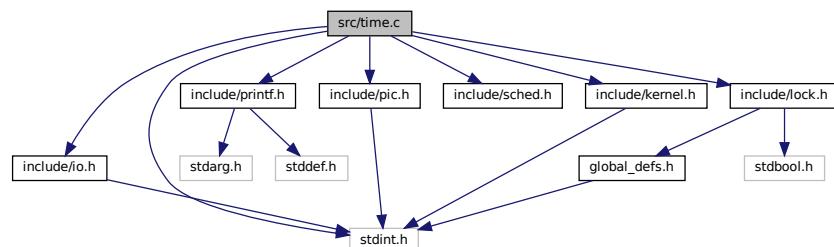
```

```

01040     case CHARSET_DEFAULT:
01041         break;
01042     case CHARSET_DEC_SPECIAL:
01043         c = dec_special_to_cp437(c);
01044     }
01045
01046     ctx->raw_putchar(ctx, c);
01047 }
```

## 4.139 src/time.c File Reference

```
#include "include/io.h"
#include "include/printf.h"
#include "include/pic.h"
#include "include/sched.h"
#include "include/kernel.h"
#include "include/lock.h"
#include <stdint.h>
Include dependency graph for time.c:
```



## Macros

- #define CHANNEL\_ZERO 0x40
- #define MODE\_CMD 0x43
- #define CURRENT\_YEAR 2022
- #define CMOS\_ADDR 0x70
- #define CMOS\_DATA 0x71

## Functions

- void config\_PIT (uint8\_t freq)
- void timer\_irq ()
- void sleep (uint64\_t millis)
- void sys\_clock\_handler ()
- void task\_switch\_handler ()
- void delta\_int (uint64\_t delta, uint8\_t vector)
- void init\_PIT ()
- int get\_update\_flag ()
- uint8\_t get\_RTC\_register (int reg)
- void read\_rtc ()
- void print\_date ()
- dayofweek (y, m, d)
- void print\_sys\_time ()

## Variables

- int8\_t `century_register` = 0x00
- uint8\_t `second`
- uint8\_t `minute`
- uint8\_t `hour`
- uint8\_t `day`
- uint8\_t `month`
- uint8\_t `year`
- uint64\_t `system_timer_ms` = 0
- uint64\_t `system_timer_fractions` = 0
- volatile uint64\_t `count_down`
- uint64\_t `pit_armed` = 0
- static const char \* `weekday` []
- static const char \* `month_str` []
- uint8\_t `task_timer_count` = 0

### 4.139.1 Macro Definition Documentation

#### 4.139.1.1 CHANNEL\_ZERO

```
#define CHANNEL_ZERO 0x40
```

Definition at line 9 of file [time.c](#).

#### 4.139.1.2 CMOS\_ADDR

```
#define CMOS_ADDR 0x70
```

Definition at line 12 of file [time.c](#).

#### 4.139.1.3 CMOS\_DATA

```
#define CMOS_DATA 0x71
```

Definition at line 13 of file [time.c](#).

#### 4.139.1.4 CURRENT\_YEAR

```
#define CURRENT_YEAR 2022
```

Definition at line 11 of file [time.c](#).

#### 4.139.1.5 MODE\_CMD

```
#define MODE_CMD 0x43
```

Definition at line 10 of file [time.c](#).

### 4.139.2 Function Documentation

#### 4.139.2.1 config\_PIT()

```
void config_PIT (
    uint8_t freq )
```

Here is the caller graph for this function:

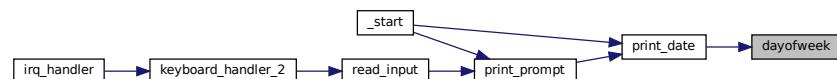


#### 4.139.2.2 dayofweek()

```
dayofweek (
    Y ,
    m ,
    d )
```

Definition at line 249 of file [time.c](#).

Here is the caller graph for this function:



#### 4.139.2.3 delta\_int()

```
void delta_int (
    uint64_t delta,
    uint8_t vector )
```

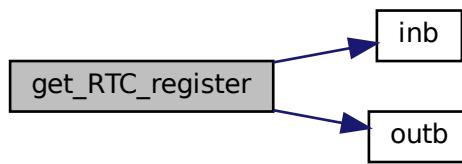
Definition at line 100 of file [time.c](#).

#### 4.139.2.4 get\_RTC\_register()

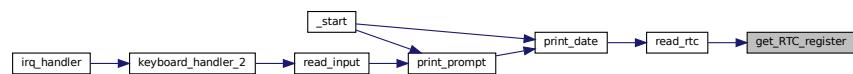
```
uint8_t get_RTC_register (
    int reg )
```

Definition at line 123 of file [time.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

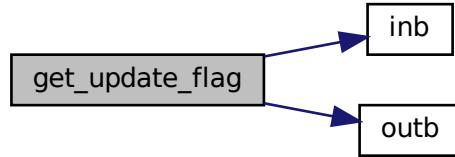


#### 4.139.2.5 get\_update\_flag()

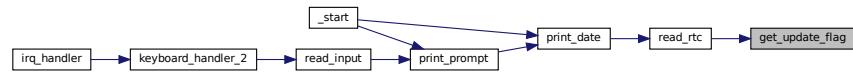
```
int get_update_flag ( )
```

Definition at line 116 of file [time.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

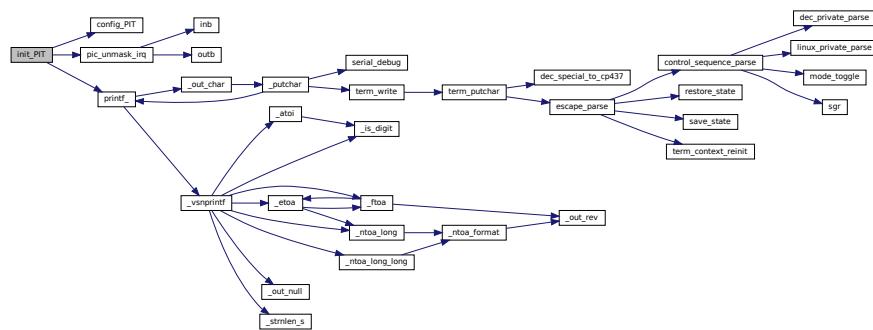


#### 4.139.2.6 init\_PIT()

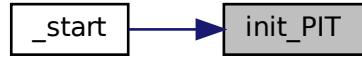
```
void init_PIT ( )
```

Definition at line 104 of file [time.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

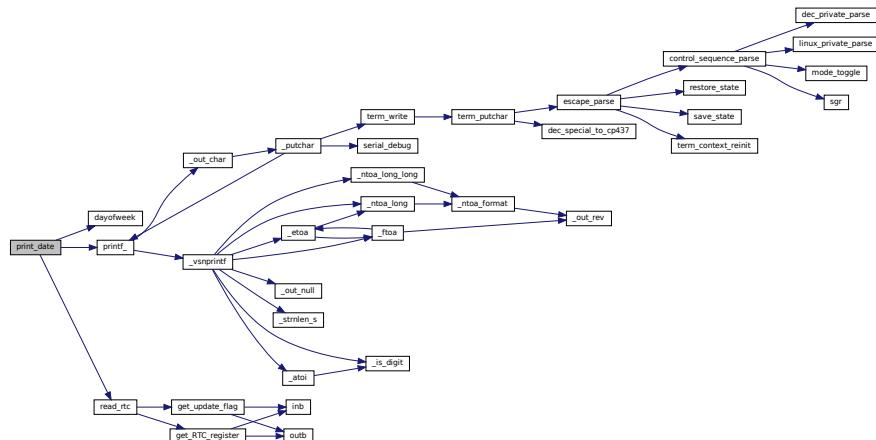


#### 4.139.2.7 print\_date()

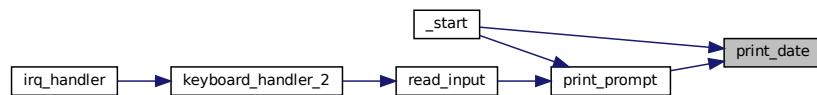
```
void print_date( )
```

Definition at line 223 of file [time.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

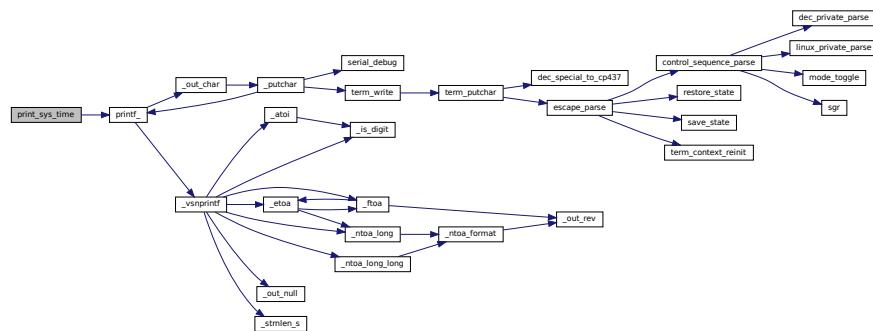


#### 4.139.2.8 print\_sys\_time()

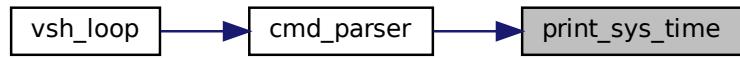
```
void print_sys_time ( )
```

Definition at line 259 of file [time.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

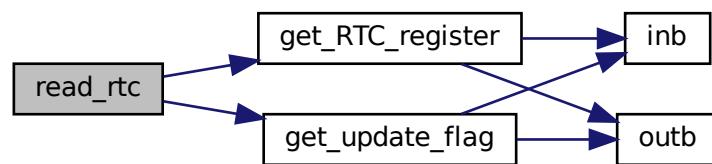


#### 4.139.2.9 read\_rtc()

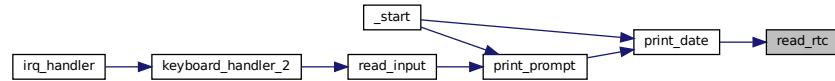
```
void read_rtc ( )
```

Definition at line 130 of file [time.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.139.2.10 sleep()

```
void sleep (
    uint64_t millis )
```

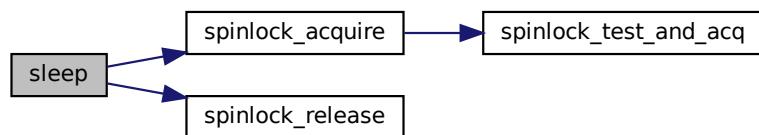
Sleep for a number of milliseconds. This is useful for unit testing to ensure that the program doesn't die indefinitely

**Parameters**

millis	- The number of milliseconds to
--------	---------------------------------

Definition at line 64 of file [time.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.139.2.11 sys\_clock\_handler()

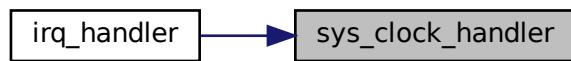
```
void sys_clock_handler ( )
```

Definition at line 81 of file [time.c](#).

Here is the call graph for this function:



Here is the caller graph for this function:

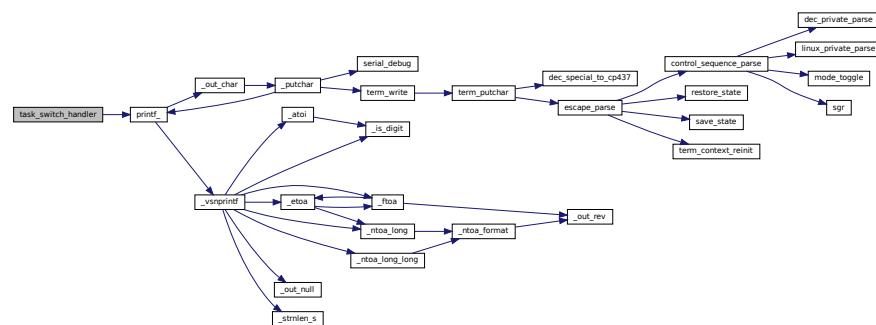


#### 4.139.2.12 task\_switch\_handler()

```
void task_switch_handler ( )
```

Definition at line 92 of file [time.c](#).

Here is the call graph for this function:



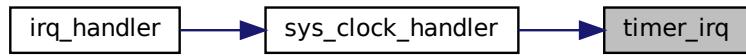
Here is the caller graph for this function:



#### 4.139.2.13 timer\_irq()

```
void timer_irq ( )
```

Here is the caller graph for this function:



### 4.139.3 Variable Documentation

#### 4.139.3.1 century\_register

```
int8_t century_register = 0x00
```

Definition at line 18 of file [time.c](#).

#### 4.139.3.2 count\_down

```
volatile uint64_t count_down
```

Definition at line 26 of file [time.c](#).

#### 4.139.3.3 day

```
uint8_t day
```

Definition at line 20 of file [time.c](#).

#### 4.139.3.4 hour

```
uint8_t hour
```

Definition at line 20 of file [time.c](#).

#### 4.139.3.5 minute

```
uint8_t minute
```

Definition at line 20 of file [time.c](#).

#### 4.139.3.6 month

```
uint8_t month
```

Definition at line 20 of file [time.c](#).

#### 4.139.3.7 month\_str

```
const char* month_str[ ] [static]
```

##### Initial value:

```
= {  
    "Jan",  
    "Feb",  
    "Mar",  
    "Apr",  
    "May",  
    "Jun",  
    "Jul",  
    "Aug",  
    "Sep",  
    "Oct",  
    "Nov",  
    "Dec"  
}
```

Definition at line 42 of file [time.c](#).

#### 4.139.3.8 pit\_armed

```
uint64_t pit_armed = 0
```

Definition at line 28 of file [time.c](#).

#### 4.139.3.9 second

```
uint8_t second
```

Definition at line 20 of file [time.c](#).

#### 4.139.3.10 system\_timer\_fractions

```
uint64_t system_timer_fractions = 0
```

Definition at line 24 of file [time.c](#).

#### 4.139.3.11 system\_timer\_ms

```
uint64_t system_timer_ms = 0
```

Definition at line 22 of file [time.c](#).

#### 4.139.3.12 task\_timer\_count

```
uint8_t task_timer_count = 0
```

Definition at line 90 of file [time.c](#).

#### 4.139.3.13 weekday

```
const char* weekday[] [static]
```

##### Initial value:

```
= {  
    "Sun",  
    "Mon",  
    "Tue",  
    "Wen",  
    "Thu",  
    "Fri",  
    "Sat"  
}
```

Definition at line 30 of file [time.c](#).

### 4.139.3.14 year

```
uint8_t year
```

Definition at line 20 of file [time.c](#).

## 4.140 time.c

```
00001 #include "include/io.h"
00002 #include "include/printf.h"
00003 #include "include/pic.h"
00004 #include "include/sched.h"
00005 #include "include/kernel.h"
00006 #include "include/lock.h"
00007 #include <stdint.h>
00008
00009 #define CHANNEL_ZERO 0x40
00010 #define MODE_CMD 0x43
00011 #define CURRENT_YEAR 2022
00012 #define CMOS_ADDR 0x70
00013 #define CMOS_DATA 0x71
00014
00015 extern void config_PIT(uint8_t freq);
00016 extern void timer_irq();
00017
00018 int8_t century_register = 0x00;
00019
00020 uint8_t second, minute, hour, day, month, year;
00021
00022 uint64_t system_timer_ms = 0;
00023
00024 uint64_t system_timer_fractions = 0;
00025
00026 volatile uint64_t count_down;
00027
00028 uint64_t pit_armed = 0;
00029
00030 static const char *weekday[] = {
00031     "Sun",
00032     "Mon",
00033     "Tue",
00034     "Wed",
00035     "Thu",
00036     "Fri",
00037     "Sat"
00038 };
00039
00040 };
00041
00042 static const char *month_str[] = {
00043     "Jan",
00044     "Feb",
00045     "Mar",
00046     "Apr",
00047     "May",
00048     "Jun",
00049     "Jul",
00050     "Aug",
00051     "Sep",
00052     "Oct",
00053     "Nov",
00054     "Dec"
00055 };
00056
00057 };
00058
00064 void sleep(uint64_t millis)
00065 {
00066     static spinlock_t sleep_lock = SPINLOCK_INIT;
00067
00068     count_down = millis;
00069
00070     spinlock_acquire(&sleep_lock);
00071
00072     while (count_down > 0)
00073     {
00074         // printf("%i\n", count_down);
00075         // halt();
00076     }
00077 }
```

```
00078     spinlock_release(&sleep_lock);
00079 }
00080
00081 void sys_clock_handler()
00082 {
00083
00084     system_timer_fractions++;
00085     system_timer_ms++;
00086
00087     timer_irq();
00088 }
00089
00090 uint8_t task_timer_count = 0;
00091
00092 void task_switch_handler()
00093 {
00094
00095     printf_("%s\n", "Switched!");
00096
00097     task_timer_count = 0;
00098 }
00099
00100 void delta_int(uint64_t delta, uint8_t vector)
00101 {
00102 }
00103
00104 void init_PIT()
00105 {
00106
00107     config_PIT(1000);
00108
00109     pic_unmask_irq(0);
00110
00111     pit_armed = 1;
00112
00113     printf_("%s\n", "PIT Online!");
00114 }
00115
00116 int get_update_flag()
00117 {
00118
00119     outb(CMOS_ADDR, 0x0A);
00120     return (inb(CMOS_DATA) & 0x80);
00121 }
00122
00123 uint8_t get_RTC_register(int reg)
00124 {
00125
00126     outb(CMOS_ADDR, reg);
00127     return inb(CMOS_DATA);
00128 }
00129
00130 void read_rtc()
00131 {
00132     unsigned char century;
00133     unsigned char last_second;
00134     unsigned char last_minute;
00135     unsigned char last_hour;
00136     unsigned char last_day;
00137     unsigned char last_month;
00138     unsigned char last_year;
00139     unsigned char last_century;
00140     unsigned char registerB;
00141
00142 // Note: This uses the "read registers until you get the same values twice in a row" technique
00143 //       to avoid getting dodgy/inconsistent values due to RTC updates
00144
00145     while (get_update_flag())
00146         ; // Make sure an update isn't in progress
00147     second = get_RTC_register(0x00);
00148     minute = get_RTC_register(0x02);
00149     hour = get_RTC_register(0x04);
00150     day = get_RTC_register(0x07);
00151     month = get_RTC_register(0x08);
00152     year = get_RTC_register(0x09);
00153     if (century_register != 0)
00154     {
00155         century = get_RTC_register(century_register);
00156     }
00157
00158     do
00159     {
00160         last_second = second;
00161         last_minute = minute;
00162         last_hour = hour;
00163         last_day = day;
00164         last_month = month;
```

```

00165     last_year = year;
00166     last_century = century;
00167
00168     while (get_update_flag())
00169         ; // Make sure an update isn't in progress
00170     second = get_RTC_register(0x00);
00171     minute = get_RTC_register(0x02);
00172     hour = get_RTC_register(0x04);
00173     day = get_RTC_register(0x07);
00174     month = get_RTC_register(0x08);
00175     year = get_RTC_register(0x09);
00176     if (century_register != 0)
00177     {
00178         century = get_RTC_register(century_register);
00179     }
00180 } while ((last_second != second) || (last_minute != minute) || (last_hour != hour) ||
00181     (last_day != day) || (last_month != month) || (last_year != year) ||
00182     (last_century != century));
00183
00184 registerB = get_RTC_register(0x0B);
00185
00186 // Convert BCD to binary values if necessary
00187
00188 if (!(registerB & 0x04))
00189 {
00190     second = (second & 0x0F) + ((second / 16) * 10);
00191     minute = (minute & 0x0F) + ((minute / 16) * 10);
00192     hour = ((hour & 0x0F) + (((hour & 0x70) / 16) * 10)) | (hour & 0x80);
00193     day = (day & 0x0F) + ((day / 16) * 10);
00194     month = (month & 0x0F) + ((month / 16) * 10);
00195     year = (year & 0x0F) + ((year / 16) * 10);
00196     if (century_register != 0)
00197     {
00198         century = (century & 0x0F) + ((century / 16) * 10);
00199     }
00200 }
00201
00202 // Convert 12 hour clock to 24 hour clock if necessary
00203
00204 if (!(registerB & 0x02) && (hour & 0x80))
00205 {
00206     hour = ((hour & 0x7F) + 12) % 24;
00207 }
00208
00209 // Calculate the full (4-digit) year
00210
00211 if (century_register != 0)
00212 {
00213     year += century * 100;
00214 }
00215 else
00216 {
00217     year += (CURRENT_YEAR / 100) * 100;
00218     if (year < CURRENT_YEAR)
00219         year += 100;
00220 }
00221 }
00222
00223 void print_date()
00224 {
00225
00226     // TODO: Work out the kinks
00227
00228     read_rtc();
00229
00230     printf_( "%s", weekday[dayofweek(year, month, day)]);
00231
00232     printf_( "%s", " ");
00233
00234     printf_( "%i", day);
00235
00236     printf_( "%s", " ");
00237
00238     printf_( "%s", month_str[month - 1]);
00239
00240     printf_( "%s", " ");
00241
00242     printf_( "%i", year);
00243
00244     printf_( "%s\n", " (VERY WIP)");
00245 }
00246
00247 // Courtesy of Tomohiko Sakamoto
00248
00249 dayofweek(y, m, d) /* 1 <= m <= 12, y > 1752 (in the U.K.) */
00250 {
00251     static int t[] = {0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4};

```

```

00252     if (m < 3)
00253     {
00254         y -= 1;
00255     }
00256     return (y + y / 4 - y / 100 + y / 400 + t[m - 1] + d) % 7;
00257 }
00258
00259 void print_sys_time()
00260 {
00261
00262     printf_( "%s", "Elapsed system time is: " );
00263     printf_( "%i", system_timer_ms );
00264     printf_( "%s", "." );
00265     printf_( "%i", system_timer_fractions );
00266     printf_( "%s\n", " ms." );
00267 }

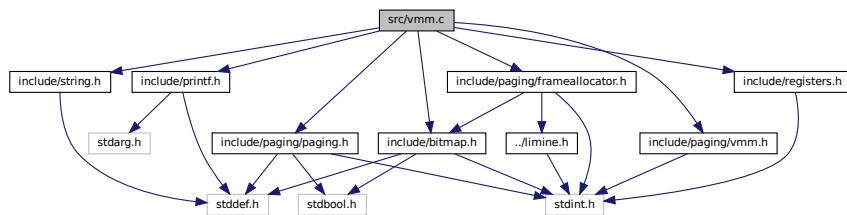
```

## 4.141 src/vmm.c File Reference

```

#include "include(bitmap.h"
#include "include/paging/frameallocator.h"
#include "include/paging/paging.h"
#include "include/paging/vmm.h"
#include "include/registers.h"
#include "include/string.h"
#include "include/printf.h"
Include dependency graph for vmm.c:

```



### Functions

- void [VMM\\_table\\_clone \(\)](#)

### Variables

- static struct PageTable \* [vmm\\_page\\_table](#)
- struct [dummy\\_proc](#) \* [test\\_proc](#)
- uint64\_t [proc\\_page\\_size](#) = 0

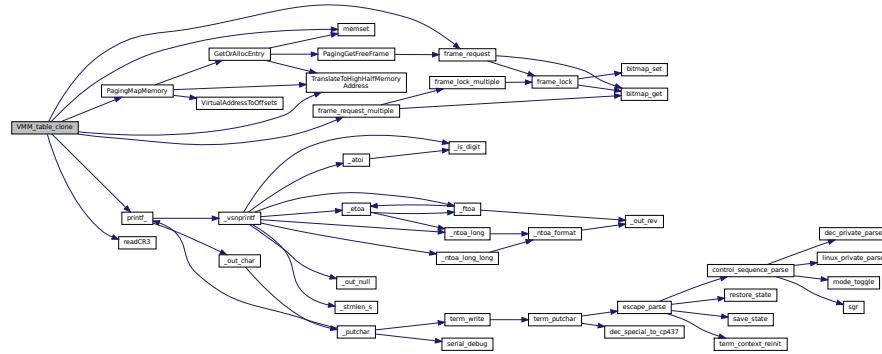
#### 4.141.1 Function Documentation

### 4.141.1.1 VMM\_table\_clone()

```
void VMM_table_clone ( )
```

Definition at line 14 of file [vmm.c](#).

Here is the call graph for this function:



## 4.141.2 Variable Documentation

### 4.141.2.1 proc\_page\_size

```
uint64_t proc_page_size = 0
```

Definition at line 12 of file [vmm.c](#).

### 4.141.2.2 test\_proc

```
struct dummy_proc* test_proc
```

Definition at line 10 of file [vmm.c](#).

### 4.141.2.3 vmm\_page\_table

```
struct PageTable* vmm_page_table [static]
```

Definition at line 9 of file [vmm.c](#).

## 4.142 vmm.c

```
00001 #include "include/bitmap.h"
00002 #include "include/paging/frameallocator.h"
00003 #include "include/paging/paging.h"
00004 #include "include/paging/vmm.h"
00005 #include "include/registers.h"
00006 #include "include/string.h"
00007 #include "include/printf.h"
00008
00009 static struct PageTable *vmm_page_table;
00010 struct dummy_proc *test_proc;
00011
00012 uint64_t proc_page_size = 0;
00013
00014 void VMM_table_clone()
00015 {
00016
00017     static struct PageTable *table_frame;
00018     static struct PageTable *current_table;
00019     static struct PageTable *higher_frame;
00020
00021     // test_proc->id = 0x0;
00022     // test_proc->data = 0x15000;
00023
00024     table_frame = (struct PageTable *)frame_request();
00025     current_table = (struct PageTable *)readCR3();
00026
00027     PagingMapMemory(current_table, TranslateToHighHalfMemoryAddress((uint64_t)table_frame),
00028                     table_frame, PAGING_FLAG_PRESENT | PAGING_FLAG_WRITABLE | PAGING_FLAG_USER_ACCESSIBLE);
00029
00030     higher_frame = (struct PageTable *)TranslateToHighHalfMemoryAddress((uint64_t)table_frame);
00031
00032     memset(higher_frame, 0, sizeof(struct PageTable));
00033
00034     auto proc_page_size = 0x8000 / 0x1000 + 1;
00035
00036     printf_(""%s\n", "pages used by test: ");
00037     printf_(""%i\n", proc_page_size);
00038
00039     test_proc = (struct dummy_proc *)frame_request_multiple(proc_page_size);
00039 }
```

# Index

- \_CPU\_UTILS\_H
  - cpuUtils.h, 145
- \_HASHMAP\_H
  - hashmap.h, 213
- \_IO\_H
  - io.h, 199
- \_KERNEL\_UTILS\_H
  - KernelUtils.h, 207
- \_KHEAPBLOCKBM, 184
- \_KHEAPBM, 184
- \_LOCK\_C\_H
  - lock.h, 254
- \_PRINTF\_H\_
  - printf.h, 290
- \_SHELL\_H
  - shell.h, 309
- \_TIME\_H
  - time.h, 329
- \_TSS\_H
  - tss.h, 337
- \_attribute\_\_\_, 190
  - paging.h, 273
- \_atoi
  - printf.c, 419
- \_etoa
  - printf.c, 420
- \_ftoa
  - printf.c, 422
- \_is\_digit
  - printf.c, 424
- \_ntoa\_format
  - printf.c, 425
- \_ntoa\_long
  - printf.c, 427
- \_ntoa\_long\_long
  - printf.c, 429
- \_out\_buffer
  - printf.c, 431
- \_out\_char
  - printf.c, 432
- \_out\_fct
  - printf.c, 433
- \_out\_null
  - printf.c, 434
- \_out\_rev
  - printf.c, 435
- \_putchar
  - printf.h, 292
  - putchar.c, 458

- \_start
- kernel.c, 351
- \_strnlen\_s
- printf.c, 436
- \_vsnprintf
- printf.c, 437
- ~ScopedLock
- ScopedLock, 46
- absorb\_right
- liballoc.c, 375
- active
- cpu\_local, 7
- ALIGN\_16BIT
- global\_defs.h, 181
- ALIGN\_4K
- global\_defs.h, 181
- ALIGN\_DOWN
- KernelUtils.c, 361
- ALIGN\_UP
- KernelUtils.c, 361
- ALLOC
- memUtils.h, 261
- alloc.cpp
- liballoc\_alloc, 60
- liballoc\_free, 60
- liballoc\_lock, 61
- liballoc\_unlock, 61
- liballocLock, 62
- allocate\_new\_tag
- liballoc.c, 375
- arg
- out\_fct\_wrap\_type, 45
- atomic\_lock
- lock.h, 255
- lock\_atm\_c.cpp, 391
- atomic\_unlock
- lock.h, 255
- lock\_atm\_c.cpp, 391
- AtomicLock, 5
- AtomicLock, 5
- ForceLock, 5
- IsLocked, 6
- Lock, 6
- Unlock, 6
- autoflush
- term\_context, 48
- bitmap.h
- bitmap\_get, 140

bitmap\_set, 141  
 bitmap\_get  
     bitmap.h, 140  
 bitmap\_set  
     bitmap.h, 141  
 bitmapSize  
     frameallocator.c, 117  
 Black  
     kernel.c, 350  
 Blue  
     kernel.c, 350  
 blue\_mask\_shift  
     limine\_framebuffer, 22  
 blue\_mask\_size  
     limine\_framebuffer, 22  
 bold  
     term\_context, 48  
 bootspace  
     kernel.c, 355  
     kernel.h, 204  
 boundary\_tag, 229  
 bpp  
     limine\_framebuffer, 22  
 breakpoint  
     gdt.c, 123  
     kernel.c, 353  
     KernelUtils.c, 362  
 BSIZE  
     memUtils.h, 261  
 bsp  
     cpu\_local, 7  
 builtin\_font  
     framebuffer.c, 490  
 c\_lock  
     lock\_atm\_c.cpp, 392  
 callback  
     term\_context, 49  
 calloc  
     liballoc.c, 376  
     liballoc.h, 229  
 CAS  
     global\_defs.h, 182  
 century\_register  
     time.c, 538  
 CHANNEL\_ZERO  
     time.c, 530  
 CHARSET\_DEC\_SPECIAL  
     term.c, 501  
 CHARSET\_DEFAULT  
     term.c, 502  
 charsets  
     term\_context, 49  
 check\_acpi  
     cpuUtils.c, 74  
 check\_apic  
     cpuUtils.c, 75  
 check\_ds  
     cpuUtils.c, 76  
 check\_fpu  
     cpuUtils.c, 76  
 check\_htt  
     cpuUtils.c, 77  
 check\_mca  
     cpuUtils.c, 78  
 check\_mce  
     cpuUtils.c, 79  
 check\_msр  
     cpuUtils.c, 79  
 check\_pae  
     cpuUtils.c, 80  
 check\_pcid  
     cpuUtils.c, 81  
 check\_sse  
     cpuUtils.c, 82  
 check\_tm  
     cpuUtils.c, 82  
 check\_vmx  
     cpuUtils.c, 83  
 check\_xsave  
     cpuUtils.c, 84  
 clear  
     term\_context, 49  
 clear\_screen  
     kernel.c, 353  
     kernel.h, 203  
 cmd\_parser  
     shell.c, 464  
 CMOS\_ADDR  
     time.c, 530  
 CMOS\_DATA  
     time.c, 530  
 col256  
     term.c, 516  
 cols  
     term\_context, 49  
 columns  
     limine\_terminal, 42  
 compare\_char  
     framebuffer.c, 473  
 config\_PIT  
     time.c, 531  
 control\_sequence  
     term\_context, 49  
 control\_sequence\_parse  
     term.c, 502  
 count\_down  
     time.c, 538  
 cpu\_ctx, 144  
 cpu\_init  
     cpuUtils.h, 150  
 cpu\_local, 7  
     active, 7  
     bsp, 7  
     cpu\_number, 8  
     idle\_thread, 8  
     last\_run\_queue\_index, 8

timer\_function, 8  
tss, 8  
cpu\_number  
  cpu\_local, 8  
CPU\_vendor  
  cpuUtils.c, 94  
  cpuUtils.h, 157  
cpudet-clean.c  
  cpuid, 64  
  detect\_cpu, 65  
  do\_amd, 65  
  do\_intel, 66  
  Intel, 67  
  Intel\_Other, 67  
  printregs, 66  
cpuid  
  cpudet-clean.c, 64  
cpuid\_check\_acpi  
  cpuUtils.c, 85  
cpuid\_check\_apic  
  cpuUtils.c, 85  
cpuid\_check\_avx  
  cpuUtils.c, 85  
cpuid\_check\_ds  
  cpuUtils.c, 85  
cpuid\_check\_fpu  
  cpuUtils.c, 86  
cpuid\_check\_fxsr  
  cpuUtils.c, 86  
cpuid\_check\_htt  
  cpuUtils.c, 86  
cpuid\_check\_mca  
  cpuUtils.c, 86  
cpuid\_check\_mce  
  cpuUtils.c, 87  
cpuid\_check\_msr  
  cpuUtils.c, 87  
cpuid\_check\_oxsave  
  cpuUtils.c, 87  
cpuid\_check\_pae  
  cpuUtils.c, 87  
cpuid\_check\_pcid  
  cpuUtils.c, 88  
cpuid\_check\_sse  
  cpuUtils.c, 88  
cpuid\_check\_tm  
  cpuUtils.c, 88  
cpuid\_check\_vmx  
  cpuUtils.c, 89  
cpuid\_check\_xsave  
  cpuUtils.c, 89  
cpuid\_readout  
  cpuUtils.c, 89  
  cpuUtils.h, 150  
CPUID\_VENDOR\_AMD  
  cpuUtils.h, 145  
CPUID\_VENDOR\_AO486  
  cpuUtils.h, 145  
CPUID\_VENDOR\_BHYVE  
  cpuUtils.h, 145  
CPUID\_VENDOR\_CENTAUR  
  cpuUtils.h, 145  
CPUID\_VENDOR\_CYRIX  
  cpuUtils.h, 146  
CPUID\_VENDOR\_ELBRUS  
  cpuUtils.h, 146  
CPUID\_VENDOR\_HYGON  
  cpuUtils.h, 146  
CPUID\_VENDOR\_HYPERV  
  cpuUtils.h, 146  
CPUID\_VENDOR\_INTEL  
  cpuUtils.h, 146  
CPUID\_VENDOR\_KVM  
  cpuUtils.h, 146  
CPUID\_VENDOR\_NEXGEN  
  cpuUtils.h, 147  
CPUID\_VENDOR\_NSC  
  cpuUtils.h, 147  
CPUID\_VENDOR\_OLDAMD  
  cpuUtils.h, 147  
CPUID\_VENDOR\_OLDAO486  
  cpuUtils.h, 147  
CPUID\_VENDOR\_OLDTRANSMETA  
  cpuUtils.h, 147  
CPUID\_VENDOR\_PARALLELS  
  cpuUtils.h, 147  
CPUID\_VENDOR\_PARALLELS\_ALT  
  cpuUtils.h, 148  
CPUID\_VENDOR\_QEMU  
  cpuUtils.h, 148  
CPUID\_VENDOR\_QNX  
  cpuUtils.h, 148  
CPUID\_VENDOR\_RISE  
  cpuUtils.h, 148  
CPUID\_VENDOR\_SIS  
  cpuUtils.h, 148  
CPUID\_VENDOR\_TRANSMETA  
  cpuUtils.h, 148  
CPUID\_VENDOR\_UMC  
  cpuUtils.h, 149  
CPUID\_VENDOR\_VIA  
  cpuUtils.h, 149  
CPUID\_VENDOR\_VIRTUALBOX  
  cpuUtils.h, 149  
CPUID\_VENDOR\_VMWARE  
  cpuUtils.h, 149  
CPUID\_VENDOR\_VORTEX  
  cpuUtils.h, 149  
CPUID\_VENDOR\_XEN  
  cpuUtils.h, 149  
CPUID\_VENDOR\_ZHAOXIN  
  cpuUtils.h, 150  
cpuUtils.c  
  check\_acpi, 74  
  check\_apic, 75  
  check\_ds, 76

check\_fpu, 76  
 check\_htt, 77  
 check\_mca, 78  
 check\_mce, 79  
 check\_msr, 79  
 check\_pae, 80  
 check\_pcid, 81  
 check\_sse, 82  
 check\_tm, 82  
 check\_vmx, 83  
 check\_xsave, 84  
 CPU\_vendor, 94  
 cpuid\_check\_acpi, 85  
 cpuid\_check\_apic, 85  
 cpuid\_check\_avx, 85  
 cpuid\_check\_ds, 85  
 cpuid\_check\_fpu, 86  
 cpuid\_check\_fxsr, 86  
 cpuid\_check\_htt, 86  
 cpuid\_check\_mca, 86  
 cpuid\_check\_mce, 87  
 cpuid\_check\_msр, 87  
 cpuid\_check\_oxsave, 87  
 cpuid\_check\_pae, 87  
 cpuid\_check\_pcid, 88  
 cpuid\_check\_sse, 88  
 cpuid\_check\_tm, 88  
 cpuid\_check\_vmx, 89  
 cpuid\_check\_xsave, 89  
 cpuid\_readout, 89  
 first\_run, 94  
 fpu\_init, 90  
 get\_model, 91  
 get\_vendor, 91  
 halt, 92  
 no\_sse, 92  
 no\_xsave, 93  
 vendor\_str1, 93  
 vendor\_str2, 94  
 vendor\_str3, 94  
 cpuUtils.h  
   \_CPU\_UTILS\_H, 145  
 cpu\_init, 150  
 CPU\_vendor, 157  
 cpuid\_readout, 150  
 CPUID\_VENDOR\_AMD, 145  
 CPUID\_VENDOR\_AO486, 145  
 CPUID\_VENDOR\_BHYVE, 145  
 CPUID\_VENDOR\_CENTAUR, 145  
 CPUID\_VENDOR\_CYRIX, 146  
 CPUID\_VENDOR\_ELBRUS, 146  
 CPUID\_VENDOR\_HYGON, 146  
 CPUID\_VENDOR\_HYPERV, 146  
 CPUID\_VENDOR\_INTEL, 146  
 CPUID\_VENDOR\_KVM, 146  
 CPUID\_VENDOR\_NEXGEN, 147  
 CPUID\_VENDOR\_NSC, 147  
 CPUID\_VENDOR\_OLDAMD, 147  
 CPUID\_VENDOR\_OLDAO486, 147  
 CPUID\_VENDOR\_OLDTRANSMETA, 147  
 CPUID\_VENDOR\_PARALLELS, 147  
 CPUID\_VENDOR\_PARALLELS\_ALT, 148  
 CPUID\_VENDOR\_QEMU, 148  
 CPUID\_VENDOR\_QNX, 148  
 CPUID\_VENDOR\_RISE, 148  
 CPUID\_VENDOR\_SIS, 148  
 CPUID\_VENDOR\_TRANSMETA, 148  
 CPUID\_VENDOR\_UMC, 149  
 CPUID\_VENDOR\_VIA, 149  
 CPUID\_VENDOR\_VIRTUALBOX, 149  
 CPUID\_VENDOR\_VMWARE, 149  
 CPUID\_VENDOR\_VORTEX, 149  
 CPUID\_VENDOR\_XEN, 149  
 CPUID\_VENDOR\_ZHAOXIN, 150  
 detect\_cpu, 151  
 fpu\_bank\_size, 158  
 fpu\_rest, 158  
 fpu\_save, 158  
 fxrstor, 152  
 fxsave, 152  
 get\_fs\_base, 152  
 get\_gs\_base, 153  
 get\_kernel\_gs\_base, 153  
 get\_model, 154  
 interrupt\_state, 154  
 rdmsr, 154  
 set\_fs\_base, 155  
 set\_gs\_base, 155  
 set\_kernel\_gs\_base, 156  
 sysenter, 158  
 this\_cpu, 156  
 wrmsr, 156  
 xrstor, 157  
 xsave, 157  
 csi  
   term\_context, 49  
 current\_charset  
   term\_context, 50  
 current\_primary  
   term\_context, 50  
 CURRENT\_YEAR  
   time.c, 530  
 Cyan  
   kernel.c, 350  
 data\_end\_addr  
   KernelUtils.c, 366  
 data\_start\_addr  
   KernelUtils.c, 366  
 day  
   time.c, 538  
   time.h, 334  
 dayofweek  
   time.c, 531  
 DC  
   gdt.h, 178  
 dec\_private

term\_context, 50  
dec\_private\_parse  
    term.c, 503  
dec\_special\_to\_cp437  
    term.c, 504  
deinit  
    term\_context, 50  
delta\_int  
    time.c, 531  
desc  
    gdt.c, 124  
detect\_cpu  
    cpudet-clean.c, 65  
    cpuUtils.h, 151  
disable\_cursor  
    term\_context, 50  
discard\_next  
    term\_context, 50  
do\_amd  
    cpudet-clean.c, 65  
do\_intel  
    cpudet-clean.c, 66  
double\_buffer\_flush  
    term\_context, 51  
draw\_cursor  
    framebuffer.c, 473  
dummy\_proc, 280  
DuplicateRecursive  
    paging.c, 400  
  
early\_term  
    kernel.c, 356  
    kernel.h, 204  
edid\_size  
    limine\_framebuffer, 22  
enable\_cursor  
    term\_context, 51  
entry\_count  
    limine\_memmap\_response, 32  
esc\_values  
    term\_context, 51  
esc\_values\_i  
    term\_context, 51  
escape  
    term\_context, 51  
escape\_offset  
    term\_context, 51  
escape\_parse  
    term.c, 505  
exception\_messages  
    interrupts.c, 344  
Execute  
    gdt.h, 178  
  
fbr\_req  
    KernelUtils.c, 366  
    KernelUtils.h, 210  
fbterm\_char, 316  
fbterm\_clear  
    framebuffer.c, 474  
fbterm\_context, 317  
fbterm\_deinit  
    framebuffer.c, 475  
fbterm\_disable\_cursor  
    framebuffer.c, 475  
fbterm\_double\_buffer\_flush  
    framebuffer.c, 475  
fbterm\_enable\_cursor  
    framebuffer.c, 476  
FBTERM\_FONT\_GLYPHS  
    framebuffer.h, 318  
fbterm\_full\_refresh  
    framebuffer.c, 476  
fbterm\_get\_cursor\_pos  
    framebuffer.c, 477  
fbterm\_init  
    framebuffer.c, 477  
    framebuffer.h, 318  
fbterm\_move\_character  
    framebuffer.c, 479  
fbterm\_queue\_item, 316  
fbterm\_raw\_putchar  
    framebuffer.c, 480  
fbterm\_restore\_state  
    framebuffer.c, 481  
fbterm\_revscroll  
    framebuffer.c, 481  
fbterm\_save\_state  
    framebuffer.c, 481  
fbterm\_scroll  
    framebuffer.c, 482  
fbterm\_set\_cursor\_pos  
    framebuffer.c, 482  
fbterm\_set\_text\_bg  
    framebuffer.c, 483  
fbterm\_set\_text\_bg\_bright  
    framebuffer.c, 483  
fbterm\_set\_text\_bg\_default  
    framebuffer.c, 484  
fbterm\_set\_text\_bg\_rgb  
    framebuffer.c, 484  
fbterm\_set\_text\_fg  
    framebuffer.c, 485  
fbterm\_set\_text\_fg\_bright  
    framebuffer.c, 485  
fbterm\_set\_text\_fg\_default  
    framebuffer.c, 486  
fbterm\_set\_text\_fg\_rgb  
    framebuffer.c, 486  
fbterm\_swap\_palette  
    framebuffer.c, 487  
fct  
    out\_fct\_wrap\_type, 45  
fctprintf  
    printf.c, 439  
    printf.h, 294  
first\_run

cpuUtils.c, 94  
**flags**  
 limine\_smp\_request, 39  
**FLAGS\_ADAPTER\_EXP**  
 printf.c, 415  
**FLAGS\_CHAR**  
 printf.c, 415  
**FLAGS\_HASH**  
 printf.c, 415  
**FLAGS\_LEFT**  
 printf.c, 416  
**FLAGS\_LONG**  
 printf.c, 416  
**FLAGS\_LONG\_LONG**  
 printf.c, 416  
**FLAGS\_PLUS**  
 printf.c, 416  
**FLAGS\_PRECISION**  
 printf.c, 416  
**FLAGS\_SHORT**  
 printf.c, 416  
**FLAGS\_SPACE**  
 printf.c, 417  
**FLAGS\_UPPERCASE**  
 printf.c, 417  
**FLAGS\_ZEROPAD**  
 printf.c, 417  
**FONT\_BYTES**  
 framebuffer.c, 473  
**ForceLock**  
 AtomicLock, 5  
**fpu\_bank\_size**  
 cpuUtils.h, 158  
**fpu\_init**  
 cpuUtils.c, 90  
**fpu\_rest**  
 cpuUtils.h, 158  
**fpu\_save**  
 cpuUtils.h, 158  
**FRAME\_EXPORT**  
 frameallocator.h, 263  
**frame\_free**  
 frameallocator.c, 110  
 frameallocator.h, 263  
**frame\_free\_multiple**  
 frameallocator.c, 111  
 frameallocator.h, 264  
**frame\_lock**  
 frameallocator.c, 112  
 frameallocator.h, 265  
**frame\_lock\_multiple**  
 frameallocator.c, 112  
 frameallocator.h, 265  
**frame\_request**  
 frameallocator.c, 113  
 frameallocator.h, 266  
**frame\_request\_multiple**  
 frameallocator.c, 114  
**frameallocator.h**, 267  
**frameallocator.c**  
 bitmapSize, 117  
 frame\_free, 110  
 frame\_free\_multiple, 111  
 frame\_lock, 112  
 frame\_lock\_multiple, 112  
 frame\_request, 113  
 frame\_request\_multiple, 114  
**frameBitmap**, 117  
 free\_ram, 114  
 freeMemory, 118  
 halt, 115  
 initialized, 118  
 MemoryMapTypeString, 115  
**print\_frame\_bitmap**, 115  
**read\_memory\_map**, 116  
**reserved\_ram**, 116  
**reservedMemory**, 118  
**used\_ram**, 117  
**usedMemory**, 118  
**frameallocator.h**  
 FRAME\_EXPORT, 263  
 frame\_free, 263  
 frame\_free\_multiple, 264  
 frame\_lock, 265  
 frame\_lock\_multiple, 265  
 frame\_request, 266  
 frame\_request\_multiple, 267  
 free\_ram, 267  
 PAGE\_SIZE, 263  
**print\_frame\_bitmap**, 268  
**read\_memory\_map**, 268  
**reserved\_ram**, 269  
**used\_ram**, 269  
**frameBitmap**  
 frameallocator.c, 117  
 KernelUtils.c, 366  
**framebuffer.c**  
 builtin\_font, 490  
 compare\_char, 473  
 draw\_cursor, 473  
 fbterm\_clear, 474  
 fbterm\_deinit, 475  
 fbterm\_disable\_cursor, 475  
 fbterm\_double\_buffer\_flush, 475  
 fbterm\_enable\_cursor, 476  
 fbterm\_full\_refresh, 476  
 fbterm\_get\_cursor\_pos, 477  
 fbterm\_init, 477  
 fbterm\_move\_character, 479  
 fbterm\_raw\_putchar, 480  
 fbterm\_restore\_state, 481  
 fbterm\_revscroll, 481  
 fbterm\_save\_state, 481  
 fbterm\_scroll, 482  
 fbterm\_set\_cursor\_pos, 482  
 fbterm\_set\_text\_bg, 483

fbterm\_set\_text\_bg\_bright, 483  
fbterm\_set\_text\_bg\_default, 484  
fbterm\_set\_text\_bg\_rgb, 484  
fbterm\_set\_text\_fg, 485  
fbterm\_set\_text\_fg\_bright, 485  
fbterm\_set\_text\_fg\_default, 486  
fbterm\_set\_text\_fg\_rgb, 486  
fbterm\_swap\_palette, 487  
FONT\_BYTES, 473  
memcpy, 487  
memset, 488  
plot\_char, 488  
plot\_char\_fast, 489  
push\_to\_queue, 489  
framebuffer.h  
    FBTERM\_FONT\_GLYPHS, 318  
    fbterm\_init, 318  
framebuffer\_count  
    limine\_framebuffer\_response, 26  
free  
    liballoc.c, 376  
    liballoc.h, 230  
free\_ram  
    frameallocator.c, 114  
    frameallocator.h, 267  
freeMemory  
    frameallocator.c, 118  
full\_refresh  
    term\_context, 52  
fxrstor  
    cpuUtils.h, 152  
fxsave  
    cpuUtils.h, 152  
g\_select  
    term\_context, 52  
GDT, 175  
gdt  
    gdt.c, 125  
gdt.c  
    breakpoint, 123  
    desc, 124  
    gdt, 125  
    halt, 123  
    ist1Stack, 125  
    ist2Stack, 125  
    LoadGDT\_Stage1, 123  
    rsp0, 125  
    serial\_debug, 123  
    tss, 125  
    TssStack, 125  
gdt.h  
    DC, 178  
    Execute, 178  
    GDTAccess16Code, 176  
    GDTAccess16Data, 176  
    GDTAccess32Code, 176  
    GDTAccess32Data, 176  
    GDTAccessDPL, 176  
    GDTAccessFlag, 178  
    GDTAccessKernelCode, 177  
    GDTAccessKernelData, 177  
    GDTAccessUserCode, 177  
    GDTAccessUserData, 177  
    GDTKernelBaseSelector, 177  
    GDTTSSSegment, 177  
    GDTUserBaseSelector, 178  
    LoadGDT\_Stage1, 178  
    Present, 178  
    ReadWrite, 178  
    rsp0, 179  
    Segments, 178  
    GDT\_Desc, 174  
    GDT\_Entry, 174  
    GDT\_Entry\_16, 174  
    GDT\_Entry\_32, 174  
    GDTAccess16Code  
        gdt.h, 176  
    GDTAccess16Data  
        gdt.h, 176  
    GDTAccess32Code  
        gdt.h, 176  
    GDTAccess32Data  
        gdt.h, 176  
    GDTAccessDPL  
        gdt.h, 176  
    GDTAccessFlag  
        gdt.h, 178  
    GDTAccessKernelCode  
        gdt.h, 177  
    GDTAccessKernelData  
        gdt.h, 177  
    GDTAccessUserCode  
        gdt.h, 177  
    GDTAccessUserData  
        gdt.h, 177  
    GDTKernelBaseSelector  
        gdt.h, 177  
    GDTTSSSegment  
        gdt.h, 177  
    GDTUserBaseSelector  
        gdt.h, 178  
    get\_cursor\_pos  
        term\_context, 52  
    get\_fs\_base  
        cpuUtils.h, 152  
    get\_gs\_base  
        cpuUtils.h, 153  
    get\_kernel\_gs\_base  
        cpuUtils.h, 153  
    get\_memory\_size  
        KernelUtils.c, 362  
        KernelUtils.h, 207  
    get\_model  
        cpuUtils.c, 91  
        cpuUtils.h, 154  
    get\_RTC\_register

time.c, 532  
 get\_update\_flag  
     time.c, 532  
 get\_vendor  
     cpuUtils.c, 91  
 getexp  
     liballoc.c, 377  
 GetOrAllocEntry  
     paging.c, 400  
 GetOrNullifyEntry  
     paging.c, 401  
 global\_defs.h  
     ALIGN\_16BIT, 181  
     ALIGN\_4K, 181  
     CAS, 182  
     mode\_t, 182  
     PACKED, 182  
 gpt\_disk\_uuid  
     limine\_file, 19  
 gpt\_part\_uuid  
     limine\_file, 19  
 Green  
     kernel.c, 350  
 green\_mask\_shift  
     limine\_framebuffer, 23  
 green\_mask\_size  
     limine\_framebuffer, 23  
  
 halt  
     cpuUtils.c, 92  
     frameallocator.c, 115  
     gdt.c, 123  
     kernel.c, 354  
     KernelUtils.c, 362  
  
 hash  
     hashmap.h, 216  
 hashmap.h  
     \_HASHMAP\_H, 213  
     hash, 216  
     HASHMAP\_DELETE, 213  
     HASHMAP\_GET, 213  
     HASHMAP\_INIT, 214  
     HASHMAP\_INSERT, 214  
     HASHMAP\_KEY\_DATA\_MAX, 214  
     HASHMAP\_REMOVE, 214  
     HASHMAP\_SGET, 214  
     HASHMAP\_SINSERT, 215  
     HASHMAP\_SREMOVE, 215  
     HASHMAP\_TYPE, 215  
  
 HASHMAP\_DELETE  
     hashmap.h, 213  
  
 HASHMAP\_GET  
     hashmap.h, 213  
  
 HASHMAP\_INIT  
     hashmap.h, 214  
  
 HASHMAP\_INSERT  
     hashmap.h, 214  
  
 HASHMAP\_KEY\_DATA\_MAX  
     hashmap.h, 214  
  
 HASHMAP\_REMOVE  
     hashmap.h, 214  
  
 HASHMAP\_SGET  
     hashmap.h, 214  
  
 HASHMAP\_SINSERT  
     hashmap.h, 215  
  
 HASHMAP\_SREMOVE  
     hashmap.h, 215  
  
 HASHMAP\_TYPE  
     hashmap.h, 215  
  
 HASHMAP\_REMOVE  
     hashmap.h, 214  
  
 HASHMAP\_SGET  
     hashmap.h, 214  
  
 HASHMAP\_SINSERT  
     hashmap.h, 215  
  
 HASHMAP\_SREMOVE  
     hashmap.h, 215  
  
 HASHMAP\_TYPE  
     hashmap.h, 215  
  
 heap.c  
     k\_heapBMAAddBlock, 129  
     k\_heapBMAAddBlockEx, 129  
     k\_heapBMAAlloc, 130  
     k\_heapBMAAllocBound, 130  
     k\_heapBMFree, 131  
     k\_heapBMGetBMSize, 131  
     k\_heapBMGetNID, 131  
     k\_heapBMInit, 132  
     k\_heapBMSet, 132  
  
 heap.h  
     k\_heapBMAAddBlock, 185  
     k\_heapBMAAddBlockEx, 186  
     k\_heapBMAAlloc, 186  
     k\_heapBMAAllocBound, 187  
     k\_heapBMFree, 187  
     k\_heapBMGetBMSize, 188  
     k\_heapBMInit, 188  
     k\_heapBMSet, 188  
     KHEAPBLOCKBM, 185  
     KHEAPBM, 185  
  
 height  
     limine\_framebuffer, 23  
  
 HIGHER\_HALF\_KERNEL\_MEMORY\_OFFSET  
     paging.h, 272  
  
 HIGHER\_HALF\_MEMORY\_OFFSET  
     paging.h, 272  
  
 hour  
     time.c, 539  
     time.h, 334  
  
 id  
     limine\_5\_level\_paging\_request, 9  
     limine\_boot\_time\_request, 10  
     limine\_bootloader\_info\_request, 11  
     limine\_dtb\_request, 13  
     limine\_efi\_system\_table\_request, 15  
     limine\_entry\_point\_request, 17  
     limine\_framebuffer\_request, 25  
     limine\_hhdm\_request, 27  
     limine\_kernel\_address\_request, 28  
     limine\_kernel\_file\_request, 29  
     limine\_memmap\_request, 31  
     limine\_module\_request, 33  
     limine\_rsdp\_request, 35  
     limine\_smbios\_request, 37  
     limine\_smp\_request, 39  
     limine\_stack\_size\_request, 40  
     limine\_terminal\_request, 43

idle\_thread  
    cpu\_local, 8  
idt  
    idt.c, 138  
idt.c  
    idt, 138  
    idt\_allocate\_vector, 136  
    idt\_free\_vector, 136  
    idt\_init, 137  
    idt\_set\_descriptor, 137  
    idtr, 138  
    isr\_stub\_table, 138  
    vectors, 138  
idt.h  
    idt\_allocate\_vector, 193  
    IDT\_CPU\_EXCEPTION\_COUNT, 191  
    IDT\_DESCRIPTOR\_CALL, 191  
    IDT\_DESCRIPTOR\_EXCEPTION, 191  
    IDT\_DESCRIPTOR\_EXTERNAL, 191  
    IDT\_DESCRIPTOR\_PRESENT, 191  
    IDT\_DESCRIPTOR\_RING1, 192  
    IDT\_DESCRIPTOR\_RING2, 192  
    IDT\_DESCRIPTOR\_RING3, 192  
    IDT\_DESCRIPTOR\_X16\_INTERRUPT, 192  
    IDT\_DESCRIPTOR\_X16\_TRAP, 192  
    IDT\_DESCRIPTOR\_X32\_INTERRUPT, 192  
    IDT\_DESCRIPTOR\_X32\_TASK, 193  
    IDT\_DESCRIPTOR\_X32\_TRAP, 193  
    idt\_free\_vector, 193  
    IDT\_HDW\_INTERRUPT\_COUNT, 193  
    idt\_init, 194  
    IDT\_MAX\_DESCRIPTOROS, 193  
    idt\_reload, 194  
    idt\_set\_descriptor, 195  
idt\_allocate\_vector  
    idt.c, 136  
    idt.h, 193  
IDT\_CPU\_EXCEPTION\_COUNT  
    idt.h, 191  
IDT\_DESCRIPTOR\_CALL  
    idt.h, 191  
IDT\_DESCRIPTOR\_EXCEPTION  
    idt.h, 191  
IDT\_DESCRIPTOR\_EXTERNAL  
    idt.h, 191  
IDT\_DESCRIPTOR\_PRESENT  
    idt.h, 191  
IDT\_DESCRIPTOR\_RING1  
    idt.h, 192  
IDT\_DESCRIPTOR\_RING2  
    idt.h, 192  
IDT\_DESCRIPTOR\_RING3  
    idt.h, 192  
IDT\_DESCRIPTOR\_X16\_INTERRUPT  
    idt.h, 192  
IDT\_DESCRIPTOR\_X16\_TRAP  
    idt.h, 192  
IDT\_DESCRIPTOR\_X32\_INTERRUPT

outb, 201  
io\_wait  
    io.c, 347  
    io.h, 200  
irq\_handler  
    interrupts.c, 342  
irq\_messages  
    interrupts.c, 344  
IsHigherHalf  
    paging.h, 273  
IsKernelHigherHalf  
    paging.h, 273  
IsLocked  
    AtomicLock, 6  
isr\_exception\_handler  
    interrupts.c, 343  
isr\_stub\_table  
    idt.c, 138  
isr\_xframe\_t, 197  
isr\_xframe\_t.base\_frame, 198  
isr\_xframe\_t.control\_registers, 197  
isr\_xframe\_t.general\_registers, 198  
ist1Stack  
    gdt.c, 125  
ist2Stack  
    gdt.c, 125  
  
k\_char  
    keyboard.c, 106  
k\_getchar  
    keyboard.c, 100  
    keyboard.h, 162  
k\_heapBMAddBlock  
    heap.c, 129  
    heap.h, 185  
k\_heapBMAddBlockEx  
    heap.c, 129  
    heap.h, 186  
k\_heapBMAAlloc  
    heap.c, 130  
    heap.h, 186  
k\_heapBMAAllocBound  
    heap.c, 130  
    heap.h, 187  
k\_heapBMFree  
    heap.c, 131  
    heap.h, 187  
k\_heapBMGetBMSize  
    heap.c, 131  
    heap.h, 188  
k\_heapBMGetNID  
    heap.c, 131  
k\_heapBMInit  
    heap.c, 132  
    heap.h, 188  
k\_heapBMSet  
    heap.c, 132  
    heap.h, 188  
k\_mode  
    KernelUtils.c, 366  
    KernelUtils.h, 210  
Kaddress\_req  
    kernel.c, 356  
    KernelUtils.c, 367  
kbd\_pop  
    keyboard.c, 100  
    keyboard.h, 162  
kbd\_push  
    keyboard.c, 101  
kbd\_stack  
    keyboard.c, 106  
KBD\_STACK\_SIZE  
    keyboard.c, 100  
kbd\_top  
    keyboard.c, 106  
kernel.c  
    \_start, 351  
    Black, 350  
    Blue, 350  
    bootspace, 355  
    breakpoint, 353  
    clear\_screen, 353  
    Cyan, 350  
    early\_term, 356  
    Green, 350  
    halt, 354  
    Kaddress\_req, 356  
    kerror\_mode, 356  
    kheap, 356  
    print\_prompt, 354  
    Purple, 350  
    Red, 351  
    start\_interrupts, 355  
    stop\_interrupts, 355  
    task\_switch\_int, 355  
    term\_bg, 356  
    term\_context, 357  
    term\_fg, 357  
    test\_table, 357  
    White, 351  
    Yellow, 351  
kernel.h  
    bootspace, 204  
    clear\_screen, 203  
    early\_term, 204  
    kerror\_mode, 205  
    print\_prompt, 203  
KernelUtils.c  
    ALIGN\_DOWN, 361  
    ALIGN\_UP, 361  
    breakpoint, 362  
    data\_end\_addr, 366  
    data\_start\_addr, 366  
    fbr\_req, 366  
    frameBitmap, 366  
    get\_memory\_size, 362  
    halt, 362

init\_memory, 363  
k\_mode, 366  
Kaddress\_req, 367  
memmap\_req, 367  
page\_table, 367  
print\_memmap, 363  
print\_memory, 364  
rodata\_end\_addr, 367  
rodata\_start\_addr, 367  
smp\_req, 368  
symbol, 361  
system\_readout, 365  
text\_end\_addr, 368  
text\_start\_addr, 368  
KernelUtils.h  
  \_\_KERNEL\_UTILS\_H, 207  
  fbr\_req, 210  
  get\_memory\_size, 207  
  init\_memory, 207  
  k\_mode, 210  
  memmap\_req, 210  
  print\_memmap, 208  
  print\_memory, 209  
  system\_readout, 209  
  term\_context, 211  
kerror\_mode  
  kernel.c, 356  
  kernel.h, 205  
keyboard.c  
  k\_char, 106  
  k\_getchar, 100  
  kbd\_pop, 100  
  kbd\_push, 101  
  kbd\_stack, 106  
  KBD\_STACK\_SIZE, 100  
  kbd\_top, 106  
  keyboard\_handler, 102  
  keyboard\_handler\_2, 102  
  keyboard\_init, 104  
  read\_input, 104  
  reader\_lock, 106  
keyboard.h  
  k\_getchar, 162  
  kbd\_pop, 162  
  KEYBOARD\_DATA\_PORT, 161  
  keyboard\_handler, 162  
  keyboard\_handler\_2, 163  
  keyboard\_init, 164  
  KEYBOARD\_STATUS\_PORT, 161  
  read\_input, 165  
keyboard\_charmap  
  keyboard\_map.h, 168  
KEYBOARD\_DATA\_PORT  
  keyboard.h, 161  
keyboard\_handler  
  keyboard.c, 102  
  keyboard.h, 162  
keyboard\_handler\_2

  keyboard.c, 102  
  keyboard.h, 163  
  keyboard\_init  
    keyboard.c, 104  
    keyboard.h, 164  
  keyboard\_map  
    keyboard\_map.h, 168  
  keyboard\_map.h  
    keyboard\_charmap, 168  
    keyboard\_map, 168  
  KEYBOARD\_STATUS\_PORT  
    keyboard.h, 161  
  kheap  
    kernel.c, 356  
    memUtils.h, 261  
  KHEAPBLOCKBM  
    heap.h, 185  
  KHEAPBM  
    heap.h, 185  
  kswitches, 206  
  l\_completePages  
    liballoc.c, 382  
  l\_freePages  
    liballoc.c, 382  
  l\_initialized  
    liballoc.c, 383  
  l\_pageCount  
    liballoc.c, 383  
  lPageSize  
    liballoc.c, 383  
  last\_run\_queue\_index  
    cpu\_local, 8  
  liballoc.c  
    absorb\_right, 375  
    allocate\_new\_tag, 375  
    calloc, 376  
    free, 376  
    getexp, 377  
    insert\_tag, 378  
    l\_completePages, 382  
    l\_freePages, 382  
    l\_initialized, 383  
    l\_pageCount, 383  
    lPageSize, 383  
    LIBALLOC\_MAGIC, 374  
    liballoc\_memcpy, 378  
    liballoc\_memset, 379  
    malloc, 379  
    MAXCOMPLETE, 374  
    MAXEXP, 374  
    melt\_left, 380  
    MINEXP, 374  
    MODE, 374  
    MODE\_BEST, 374  
    MODE\_INSTANT, 375  
    realloc, 380  
    remove\_tag, 381  
    split\_tag, 382

liballoc.h  
 calloc, 229  
 free, 230  
 liballoc\_alloc, 231  
 liballoc\_free, 231  
 liballoc\_lock, 231  
 liballoc\_unlock, 232  
 malloc, 233  
 NULL, 229  
 realloc, 234  
 liballoc\_alloc  
     alloc.cpp, 60  
     liballoc.h, 231  
 liballoc\_free  
     alloc.cpp, 60  
     liballoc.h, 231  
 liballoc\_lock  
     alloc.cpp, 61  
     liballoc.h, 231  
 LIBALLOC\_MAGIC  
     liballoc.c, 374  
 liballoc\_memcpy  
     liballoc.c, 378  
 liballoc\_memset  
     liballoc.c, 379  
 liballoc\_unlock  
     alloc.cpp, 61  
     liballoc.h, 232  
 liballocLock  
     alloc.cpp, 62  
 limine.h  
     LIMINE\_5\_LEVEL\_PAGING\_REQUEST, 240  
     LIMINE\_BOOT\_TIME\_REQUEST, 240  
     LIMINE\_BOOTLOADER\_INFO\_REQUEST, 240  
     LIMINE\_COMMON\_MAGIC, 240  
     LIMINE\_DTB\_REQUEST, 240  
     LIMINE\_EFI\_SYSTEM\_TABLE\_REQUEST, 240  
 limine\_entry\_point, 247  
     LIMINE\_ENTRY\_POINT\_REQUEST, 241  
     LIMINE\_FRAMEBUFFER\_REQUEST, 241  
     LIMINE\_FRAMEBUFFER\_RGB, 241  
 limine\_goto\_address, 247  
     LIMINE\_HHDM\_REQUEST, 241  
     LIMINE\_KERNEL\_ADDRESS\_REQUEST, 241  
     LIMINE\_KERNEL\_FILE\_REQUEST, 241  
     LIMINE\_MEDIA\_TYPE\_GENERIC, 242  
     LIMINE\_MEDIA\_TYPE\_OPTICAL, 242  
     LIMINE\_MEDIA\_TYPE\_TFTP, 242  
     LIMINE\_MEMMAP\_ACPI\_NVS, 242  
     LIMINE\_MEMMAP\_ACPI\_RECLAMABLE, 242  
     LIMINE\_MEMMAP\_BAD\_MEMORY, 242  
     LIMINE\_MEMMAP\_BOOTLOADER\_RECLAMABLE, 243  
     LIMINE\_MEMMAP\_FRAMEBUFFER, 243  
     LIMINE\_MEMMAP\_KERNEL\_AND\_MODULES, 243  
     LIMINE\_MEMMAP\_REQUEST, 243  
     LIMINE\_MEMMAP\_RESERVED, 243  
 LIMINE\_MEMMAP\_USABLE, 243  
 LIMINE\_MODULE\_REQUEST, 244  
 LIMINE\_PTR, 244  
 LIMINE\_RSDP\_REQUEST, 244  
 LIMINE\_SMBIOS\_REQUEST, 244  
 LIMINE\_SMP\_REQUEST, 244  
 LIMINE\_STACK\_SIZE\_REQUEST, 244  
 limine\_terminal\_callback, 247  
 LIMINE\_TERMINAL\_CB\_BELL, 245  
 LIMINE\_TERMINAL\_CB\_DEC, 245  
 LIMINE\_TERMINAL\_CB\_KBD\_LEDS, 245  
 LIMINE\_TERMINAL\_CB\_LINUX, 245  
 LIMINE\_TERMINAL\_CB\_MODE, 245  
 LIMINE\_TERMINAL\_CB\_POS\_REPORT, 245  
 LIMINE\_TERMINAL\_CB\_PRIVATE\_ID, 246  
 LIMINE\_TERMINAL\_CB\_STATUS\_REPORT, 246  
 LIMINE\_TERMINAL\_CTX\_RESTORE, 246  
 LIMINE\_TERMINAL\_CTX\_SAVE, 246  
 LIMINE\_TERMINAL\_CTX\_SIZE, 246  
 LIMINE\_TERMINAL\_FULL\_REFRESH, 246  
 LIMINE\_TERMINAL\_REQUEST, 247  
 limine\_terminal\_write, 247  
 LIMINE\_5\_LEVEL\_PAGING\_REQUEST  
     limine.h, 240  
 limine\_5\_level.paging\_request, 9  
     id, 9  
     LIMINE\_PTR, 9  
     revision, 9  
 limine\_5\_level.paging\_response, 238  
 LIMINE\_BOOT\_TIME\_REQUEST  
     limine.h, 240  
 limine\_boot\_time\_request, 10  
     id, 10  
     LIMINE\_PTR, 10  
     revision, 10  
 limine\_boot\_time\_response, 239  
 LIMINE\_BOOTLOADER\_INFO\_REQUEST  
     limine.h, 240  
 limine\_bootloader\_info\_request, 11  
     id, 11  
     LIMINE\_PTR, 11  
     revision, 11  
 limine\_bootloader\_info\_response, 12  
     LIMINE\_PTR, 12  
     revision, 12  
 LIMINE\_COMMON\_MAGIC  
     limine.h, 240  
 LIMINE\_DTB\_REQUEST  
     limine.h, 240  
 limine\_dtb\_request, 13  
     id, 13  
     LIMINE\_PTR, 13  
     revision, 13  
 limine\_dtb\_response, 14  
     LIMINE\_PTR, 14  
     revision, 14  
 LIMINE\_EFI\_SYSTEM\_TABLE\_REQUEST  
     limine.h, 240

limine\_esi\_system\_table\_request, 14  
id, 15  
LIMINE\_PTR, 15  
revision, 15  
limine\_esi\_system\_table\_response, 16  
LIMINE\_PTR, 16  
revision, 16  
limine\_entry\_point  
limine.h, 247  
LIMINE\_ENTRY\_POINT\_REQUEST  
limine.h, 241  
limine\_entry\_point\_request, 16  
id, 17  
LIMINE\_PTR, 17  
revision, 17  
limine\_entry\_point\_response, 239  
limine\_file, 18  
gpt\_disk\_uuid, 19  
gpt\_part\_uuid, 19  
LIMINE\_PTR, 19  
mbr\_disk\_id, 19  
media\_type, 20  
part\_uuid, 20  
partition\_index, 20  
revision, 20  
size, 20  
tftp\_ip, 20  
tftp\_port, 21  
unused, 21  
limine\_framebuffer, 21  
blue\_mask\_shift, 22  
blue\_mask\_size, 22  
bpp, 22  
edid\_size, 22  
green\_mask\_shift, 23  
green\_mask\_size, 23  
height, 23  
LIMINE\_PTR, 22  
memory\_model, 23  
pitch, 23  
red\_mask\_shift, 23  
red\_mask\_size, 24  
unused, 24  
width, 24  
LIMINE\_FRAMEBUFFER\_REQUEST  
limine.h, 241  
limine\_framebuffer\_request, 24  
id, 25  
LIMINE\_PTR, 25  
revision, 25  
limine\_framebuffer\_response, 25  
framebuffer\_count, 26  
LIMINE\_PTR, 26  
revision, 26  
LIMINE\_FRAMEBUFFER\_RGB  
limine.h, 241  
limine\_goto\_address  
limine.h, 247  
LIMINE\_HHDM\_REQUEST  
limine.h, 241  
limine\_hhdm\_request, 26  
id, 27  
LIMINE\_PTR, 27  
revision, 27  
limine\_hhdm\_response, 238  
LIMINE\_KERNEL\_ADDRESS\_REQUEST  
limine.h, 241  
limine\_kernel\_address\_request, 28  
id, 28  
LIMINE\_PTR, 28  
revision, 28  
limine\_kernel\_address\_response, 239  
LIMINE\_KERNEL\_FILE\_REQUEST  
limine.h, 241  
limine\_kernel\_file\_request, 29  
id, 29  
LIMINE\_PTR, 29  
revision, 29  
limine\_kernel\_file\_response, 30  
LIMINE\_PTR, 30  
revision, 30  
LIMINE\_MEDIA\_TYPE\_GENERIC  
limine.h, 242  
LIMINE\_MEDIA\_TYPE\_OPTICAL  
limine.h, 242  
LIMINE\_MEDIA\_TYPE\_TFTP  
limine.h, 242  
LIMINE\_MEMMAP\_ACPI\_NVS  
limine.h, 242  
LIMINE\_MEMMAP\_ACPI\_RECLAIMABLE  
limine.h, 242  
LIMINE\_MEMMAP\_BAD\_MEMORY  
limine.h, 242  
LIMINE\_MEMMAP\_BOOTLOADER\_RECLAIMABLE  
limine.h, 243  
limine\_memmap\_entry, 239  
LIMINE\_MEMMAP\_FRAMEBUFFER  
limine.h, 243  
LIMINE\_MEMMAP\_KERNEL\_AND\_MODULES  
limine.h, 243  
LIMINE\_MEMMAP\_REQUEST  
limine.h, 243  
limine\_memmap\_request, 30  
id, 31  
LIMINE\_PTR, 31  
revision, 31  
LIMINE\_MEMMAP\_RESERVED  
limine.h, 243  
limine\_memmap\_response, 32  
entry\_count, 32  
LIMINE\_PTR, 32  
revision, 32  
LIMINE\_MEMMAP\_USABLE  
limine.h, 243  
LIMINE\_MODULE\_REQUEST  
limine.h, 244

limine\_module\_request, 33  
     id, 33  
     LIMINE\_PTR, 33  
     revision, 33  
 limine\_module\_response, 34  
     LIMINE\_PTR, 34  
     module\_count, 34  
     revision, 34  
**LIMINE\_PTR**  
     limine.h, 244  
     limine\_5\_level\_paging\_request, 9  
     limine\_boot\_time\_request, 10  
     limine\_bootloader\_info\_request, 11  
     limine\_bootloader\_info\_response, 12  
     limine\_dtb\_request, 13  
     limine\_dtb\_response, 14  
     limine\_efi\_system\_table\_request, 15  
     limine\_efi\_system\_table\_response, 16  
     limine\_entry\_point\_request, 17  
     limine\_file, 19  
     limine\_framebuffer, 22  
     limine\_framebuffer\_request, 25  
     limine\_framebuffer\_response, 26  
     limine\_hhdm\_request, 27  
     limine\_kernel\_address\_request, 28  
     limine\_kernel\_file\_request, 29  
     limine\_kernel\_file\_response, 30  
     limine\_memmap\_request, 31  
     limine\_memmap\_response, 32  
     limine\_module\_request, 33  
     limine\_module\_response, 34  
     limine\_rsdp\_request, 35  
     limine\_rsdp\_response, 36  
     limine\_smbios\_request, 37  
     limine\_smbios\_response, 38  
     limine\_smp\_request, 39  
     limine\_stack\_size\_request, 40  
     limine\_terminal, 42  
     limine\_terminal\_request, 43  
     limine\_terminal\_response, 44  
**LIMINE\_RSDP\_REQUEST**  
     limine.h, 244  
 limine\_rsdp\_request, 35  
     id, 35  
     LIMINE\_PTR, 35  
     revision, 35  
 limine\_rsdp\_response, 36  
     LIMINE\_PTR, 36  
     revision, 36  
**LIMINE\_SMBIOS\_REQUEST**  
     limine.h, 244  
 limine\_smbios\_request, 36  
     id, 37  
     LIMINE\_PTR, 37  
     revision, 37  
 limine\_smbios\_response, 38  
     LIMINE\_PTR, 38  
     revision, 38  
**LIMINE\_SMP\_REQUEST**  
     limine.h, 244  
 limine\_smp\_request, 39  
     flags, 39  
     id, 39  
     LIMINE\_PTR, 39  
     revision, 39  
**LIMINE\_STACK\_SIZE\_REQUEST**  
     limine.h, 244  
 limine\_stack\_size\_request, 40  
     id, 40  
     LIMINE\_PTR, 40  
     revision, 41  
     stack\_size, 41  
 limine\_stack\_size\_response, 238  
 limine\_terminal, 41  
     columns, 42  
     LIMINE\_PTR, 42  
     rows, 42  
 limine\_terminal\_callback  
     limine.h, 247  
**LIMINE\_TERMINAL\_CB\_BELL**  
     limine.h, 245  
**LIMINE\_TERMINAL\_CB\_DEC**  
     limine.h, 245  
**LIMINE\_TERMINAL\_CB\_KBD\_LEDS**  
     limine.h, 245  
**LIMINE\_TERMINAL\_CB\_LINUX**  
     limine.h, 245  
**LIMINE\_TERMINAL\_CB\_MODE**  
     limine.h, 245  
**LIMINE\_TERMINAL\_CB\_POS\_REPORT**  
     limine.h, 245  
**LIMINE\_TERMINAL\_CB\_PRIVATE\_ID**  
     limine.h, 246  
**LIMINE\_TERMINAL\_CB\_STATUS\_REPORT**  
     limine.h, 246  
**LIMINE\_TERMINAL\_CTX\_RESTORE**  
     limine.h, 246  
**LIMINE\_TERMINAL\_CTX\_SAVE**  
     limine.h, 246  
**LIMINE\_TERMINAL\_CTX\_SIZE**  
     limine.h, 246  
**LIMINE\_TERMINAL\_FULL\_REFRESH**  
     limine.h, 246  
**LIMINE\_TERMINAL\_REQUEST**  
     limine.h, 247  
 limine\_terminal\_request, 42  
     id, 43  
     LIMINE\_PTR, 43  
     revision, 43  
 limine\_terminal\_response, 43  
     LIMINE\_PTR, 44  
     revision, 44  
     terminal\_count, 44  
 limine\_terminal\_write  
     limine.h, 247  
 limine\_uuid, 238

linux\_private\_parse  
term.c, 507

LoadGDT\_Stage1  
gdt.c, 123  
gdt.h, 178

Lock  
AtomicLock, 6

lock.h  
\_LOCK\_C\_H, 254  
atomic\_lock, 255  
atomic\_unlock, 255  
smartlock\_acquire, 256  
SMARTLOCK\_INIT, 254  
smartlock\_release, 256  
spinlock\_acquire, 256  
SPINLOCK\_INIT, 254  
spinlock\_release, 257  
spinlock\_t, 255  
spinlock\_test\_and\_acq, 257

lock\_atm\_c.cpp  
atomic\_lock, 391  
atomic\_unlock, 391  
c\_lock, 392

malloc  
liballoc.c, 379  
liballoc.h, 233

MAXCOMPLETE  
liballoc.c, 374

MAXEXP  
liballoc.c, 374

mbr\_disk\_id  
limine\_file, 19

media\_type  
limine\_file, 20

melt\_left  
liballoc.c, 380

memcmp  
memcmp.c, 394  
string.h, 312

memcmp.c  
memcmp, 394

memcpy  
framebuffer.c, 487  
memcpy.c, 395  
string.h, 312

memcpy.c  
memcpy, 395

memmap\_req  
KernelUtils.c, 367  
KernelUtils.h, 210

memmove  
memmove.c, 396  
string.h, 313

memmove.c  
memmove, 396

memory\_model  
limine\_framebuffer, 23

MemoryMapTypeString

frameallocator.c, 115

memset  
framebuffer.c, 488  
memset.c, 397  
string.h, 313

memset.c  
memset, 397

memUtils.c  
PagingGetFreeFrame, 398

memUtils.h  
ALLOC, 261  
BSIZE, 261  
kheap, 261  
MEMUTILS\_H, 261  
SIZE, 261

MEMUTILS\_H  
memUtils.h, 261

MINEXP  
liballoc.c, 374

minute  
time.c, 539  
time.h, 334

MODE  
liballoc.c, 374

MODE\_BEST  
liballoc.c, 374

MODE\_CMD  
time.c, 530

MODE\_INSTANT  
liballoc.c, 375

mode\_t  
global\_defs.h, 182

mode\_toggle  
term.c, 508

module\_count  
limine\_module\_response, 34

month  
time.c, 539  
time.h, 334

month\_str  
time.c, 539

move\_character  
term\_context, 52

no\_sse  
cpuUtils.c, 92

no\_xsave  
cpuUtils.c, 93

NULL  
liballoc.h, 229

OffsetToVirtualAddress  
paging.c, 402

out\_fct\_type  
printf.c, 419

out\_fct\_wrap\_type, 45  
arg, 45  
fct, 45

outb

io.c, 348  
 io.h, 201  
**PACKED**  
 global\_defs.h, 182  
**PAGE\_ADDRESS\_MASK**  
 paging.h, 272  
**PAGE\_FLAG\_MASK**  
 paging.h, 272  
**PAGE\_SIZE**  
 frameallocator.h, 263  
**page\_table**  
 KernelUtils.c, 367  
**PageTableOffset**, 271  
**paging.c**  
 DuplicateRecursive, 400  
 GetOrAllocEntry, 400  
 GetOrNullifyEntry, 401  
 OffsetToVirtualAddress, 402  
 PagingDuplicate, 402  
 PagingIdentityMap, 402  
 PagingMapMemory, 403  
 PagingPhysicalMemory, 404  
 PagingUnmapMemory, 404  
 ReadCR3, 405  
 VirtualAddressToOffsets, 405  
 WriteCR3, 405  
**paging.h**  
 \_\_attribute\_\_, 273  
 HIGHER\_HALF\_KERNEL\_MEMORY\_OFFSET, 272  
 HIGHER\_HALF\_MEMORY\_OFFSET, 272  
 IsHigherHalf, 273  
 IsKernelHigherHalf, 273  
 PAGE\_ADDRESS\_MASK, 272  
 PAGE\_FLAG\_MASK, 272  
 PAGING\_EXPORT, 272  
 PAGING\_FLAG\_LARGER\_PAGES, 273  
 PAGING\_FLAG\_NO\_EXECUTE, 273  
 PAGING\_FLAG\_PAT0, 273  
 PAGING\_FLAG\_PAT1, 273  
 PAGING\_FLAG\_PAT2, 273  
 PAGING\_FLAG\_PRESENT, 273  
 PAGING\_FLAG\_USER\_ACCESSIBLE, 273  
 PAGING\_FLAG\_WRITABLE, 273  
 PAGING\_FLAG\_WRITE\_COMBINE, 273  
 PagingDuplicate, 273  
 PagingFlag, 272  
 PagingGetFreeFrame, 274  
 PagingIdentityMap, 274  
 PagingMapMemory, 275  
 PagingPhysicalMemory, 276  
 PagingSetActivePageTable, 276  
 PagingUnmapMemory, 276  
 TranslateToHighHalfMemoryAddress, 277  
 TranslateToKernelMemoryAddress, 277  
 TranslateToKernelPhysicalMemoryAddress, 278  
 TranslateToPhysicalMemoryAddress, 278  
**PAGING\_EXPORT**  
 paging.h, 272  
 PAGING\_FLAG\_LARGER\_PAGES  
 paging.h, 273  
 PAGING\_FLAG\_NO\_EXECUTE  
 paging.h, 273  
 PAGING\_FLAG\_PAT0  
 paging.h, 273  
 PAGING\_FLAG\_PAT1  
 paging.h, 273  
 PAGING\_FLAG\_PAT2  
 paging.h, 273  
 PAGING\_FLAG\_PRESENT  
 paging.h, 273  
 PAGING\_FLAG\_USER\_ACCESSIBLE  
 paging.h, 273  
 PAGING\_FLAG\_WRITABLE  
 paging.h, 273  
 PAGING\_FLAG\_WRITE\_COMBINE  
 paging.h, 273  
 PagingDuplicate  
 paging.c, 402  
 paging.h, 273  
 PagingFlag  
 paging.h, 272  
 PagingGetFreeFrame  
 memUtils.c, 398  
 paging.h, 274  
 PagingIdentityMap  
 paging.c, 402  
 paging.h, 274  
 PagingMapMemory  
 paging.c, 403  
 paging.h, 275  
 PagingPhysicalMemory  
 paging.c, 404  
 paging.h, 276  
 PagingSetActivePageTable  
 paging.h, 276  
 PagingUnmapMemory  
 paging.c, 404  
 paging.h, 276  
 part\_uuid  
 limine\_file, 20  
 partition\_index  
 limine\_file, 20  
**pic.c**  
 pic\_enable, 409  
 pic\_mask\_irq, 409  
 pic\_remap\_offsets, 410  
 pic\_send\_eoi, 411  
 pic\_unmask\_irq, 411  
**pic.h**  
 pic\_enable, 286  
 PIC\_EOI, 283  
 PIC\_ICW1\_ICW4, 283  
 PIC\_ICW1\_INIT, 283  
 PIC\_ICW1\_INTERVAL4, 283  
 PIC\_ICW1\_LEVEL, 283

PIC\_ICW1\_SINGLE, 284  
PIC\_ICW4\_8086, 284  
PIC\_ICW4\_AUTO, 284  
PIC\_ICW4\_BUF\_MASTER, 284  
PIC\_ICW4\_BUF\_SLAVE, 284  
pic\_mask\_irq, 286  
PIC\_MASTER, 284  
PIC\_MASTER\_COMMAND, 285  
PIC\_MASTER\_DATA, 285  
pic\_remap\_offsets, 287  
pic\_send\_eoi, 288  
PIC\_SLAVE, 285  
PIC\_SLAVE\_COMMAND, 285  
PIC\_SLAVE\_DATA, 285  
pic\_unmask\_irq, 288  
pic\_enable  
    pic.c, 409  
    pic.h, 286  
PIC\_EOI  
    pic.h, 283  
PIC\_ICW1\_ICW4  
    pic.h, 283  
PIC\_ICW1\_INIT  
    pic.h, 283  
PIC\_ICW1\_INTERVAL4  
    pic.h, 283  
PIC\_ICW1\_LEVEL  
    pic.h, 283  
PIC\_ICW1\_SINGLE  
    pic.h, 284  
PIC\_ICW4\_8086  
    pic.h, 284  
PIC\_ICW4\_AUTO  
    pic.h, 284  
PIC\_ICW4\_BUF\_MASTER  
    pic.h, 284  
PIC\_ICW4\_BUF\_SLAVE  
    pic.h, 284  
pic\_mask\_irq  
    pic.c, 409  
    pic.h, 286  
PIC\_MASTER  
    pic.h, 284  
PIC\_MASTER\_COMMAND  
    pic.h, 285  
PIC\_MASTER\_DATA  
    pic.h, 285  
pic\_remap\_offsets  
    pic.c, 410  
    pic.h, 287  
pic\_send\_eoi  
    pic.c, 411  
    pic.h, 288  
PIC\_SLAVE  
    pic.h, 285  
PIC\_SLAVE\_COMMAND  
    pic.h, 285  
PIC\_SLAVE\_DATA

    pic.h, 285  
pic\_unmask\_irq  
    pic.c, 411  
    pic.h, 288  
pit\_armed  
    time.c, 539  
    time.h, 334  
pitch  
    limine\_framebuffer, 23  
plot\_char  
    framebuffer.c, 488  
plot\_char\_fast  
    framebuffer.c, 489  
Present  
    gdt.h, 178  
print\_date  
    time.c, 534  
    time.h, 330  
print\_frame\_bitmap  
    frameallocator.c, 115  
    frameallocator.h, 268  
print\_memmap  
    KernelUtils.c, 363  
    KernelUtils.h, 208  
print\_memory  
    KernelUtils.c, 364  
    KernelUtils.h, 209  
print\_prompt  
    kernel.c, 354  
    kernel.h, 203  
print\_sys\_time  
    time.c, 534  
    time.h, 331  
printf  
    printf.h, 291  
printf.c  
    \_atoi, 419  
    \_etao, 420  
    \_ftoa, 422  
    \_is\_digit, 424  
    \_ntoa\_format, 425  
    \_ntoa\_long, 427  
    \_ntoa\_long\_long, 429  
    \_out\_buffer, 431  
    \_out\_char, 432  
    \_out\_fct, 433  
    \_out\_null, 434  
    \_out\_rev, 435  
    \_strnlen\_s, 436  
    \_vsnprintf, 437  
    fctprintf, 439  
    FLAGS\_ADAPT\_EXP, 415  
    FLAGS\_CHAR, 415  
    FLAGS\_HASH, 415  
    FLAGS\_LEFT, 416  
    FLAGS\_LONG, 416  
    FLAGS\_LONG\_LONG, 416  
    FLAGS\_PLUS, 416

FLAGS\_PRECISION, 416  
 FLAGS\_SHORT, 416  
 FLAGS\_SPACE, 417  
 FLAGS\_UPPERCASE, 417  
 FLAGS\_ZEROPAD, 417  
 out\_fct\_type, 419  
 printf\_, 440  
 PRINTF\_DEFAULT\_FLOAT\_PRECISION, 417  
 PRINTF\_FTOA\_BUFFER\_SIZE, 417  
 PRINTF\_MAX\_FLOAT, 417  
 PRINTF\_NTOA\_BUFFER\_SIZE, 418  
 PRINTF\_SUPPORT\_EXPONENTIAL, 418  
 PRINTF\_SUPPORT\_FLOAT, 418  
 PRINTF\_SUPPORT\_LONG\_LONG, 418  
 PRINTF\_SUPPORT\_PTRDIFF\_T, 418  
 snprintf\_, 442  
 sprintf\_, 443  
 vprintf\_, 443  
 vsnprintf\_, 444  
 printf.h  
     \_PRINTF\_H, 290  
     \_putchar, 292  
     fctprintf, 294  
     printf, 291  
     printf\_, 295  
     snprintf, 291  
     snprintf\_, 297  
     sprintf, 291  
     sprintf\_, 298  
     vprintf, 292  
     vprintf\_, 298  
     vsnprintf, 292  
     vsnprintf\_, 299  
 printf\_  
     printf.c, 440  
     printf.h, 295  
 PRINTF\_DEFAULT\_FLOAT\_PRECISION  
     printf.c, 417  
 PRINTF\_FTOA\_BUFFER\_SIZE  
     printf.c, 417  
 PRINTF\_MAX\_FLOAT  
     printf.c, 417  
 PRINTF\_NTOA\_BUFFER\_SIZE  
     printf.c, 418  
 PRINTF\_SUPPORT\_EXPONENTIAL  
     printf.c, 418  
 PRINTF\_SUPPORT\_FLOAT  
     printf.c, 418  
 PRINTF\_SUPPORT\_LONG\_LONG  
     printf.c, 418  
 PRINTF\_SUPPORT\_PTRDIFF\_T  
     printf.c, 418  
 printregs  
     cpudet-clean.c, 66  
 proc.h  
     proc\_yield, 302  
 proc\_page\_size  
     vmm.c, 545  
 proc\_vmm\_map, 301  
 proc\_yield  
     proc.h, 302  
 process, 302  
 Purple  
     kernel.c, 350  
 push\_to\_queue  
     framebuffer.c, 489  
 putchar.c  
     \_putchar, 458  
 raw\_putchar  
     term\_context, 53  
 rdmsr  
     cpuUtils.h, 154  
 read\_input  
     keyboard.c, 104  
     keyboard.h, 165  
 read\_memory\_map  
     frameallocator.c, 116  
     frameallocator.h, 268  
 read\_RTC  
     time.c, 535  
     time.h, 332  
 readCR2  
     registers.h, 304  
 ReadCR3  
     paging.c, 405  
 readCR3  
     registers.h, 304  
 readCR4  
     registers.h, 304  
 readCRO  
     registers.h, 304  
 reader\_lock  
     keyboard.c, 106  
 readRSP  
     registers.h, 305  
 ReadWrite  
     gdt.h, 178  
 realloc  
     liballoc.c, 380  
     liballoc.h, 234  
 Red  
     kernel.c, 351  
 red\_mask\_shift  
     limine\_framebuffer, 23  
 red\_mask\_size  
     limine\_framebuffer, 24  
 registers.c  
     writeMSR, 461  
 registers.h  
     readCR2, 304  
     readCR3, 304  
     readCR4, 304  
     readCRO, 304  
     readRSP, 305  
     writeCR0, 305  
     writeCR3, 305

writeCR4, 305  
writeMSR, 306  
remove\_tag  
  liballoc.c, 381  
reserved\_ram  
  frameallocator.c, 116  
  frameallocator.h, 269  
reservedMemory  
  frameallocator.c, 118  
restore\_state  
  term.c, 509  
  term\_context, 53  
reverse\_video  
  term\_context, 53  
revision  
  limine\_5\_level\_paging\_request, 9  
  limine\_boot\_time\_request, 10  
  limine\_bootloader\_info\_request, 11  
  limine\_bootloader\_info\_response, 12  
  limine\_dtb\_request, 13  
  limine\_dtb\_response, 14  
  limine\_efi\_system\_table\_request, 15  
  limine\_efi\_system\_table\_response, 16  
  limine\_entry\_point\_request, 17  
  limine\_file, 20  
  limine\_framebuffer\_request, 25  
  limine\_framebuffer\_response, 26  
  limine\_hhdm\_request, 27  
  limine\_kernel\_address\_request, 28  
  limine\_kernel\_file\_request, 29  
  limine\_kernel\_file\_response, 30  
  limine\_memmap\_request, 31  
  limine\_memmap\_response, 32  
  limine\_module\_request, 33  
  limine\_module\_response, 34  
  limine\_rsdp\_request, 35  
  limine\_rsdp\_response, 36  
  limine\_smbios\_request, 37  
  limine\_smbios\_response, 38  
  limine\_smp\_request, 39  
  limine\_stack\_size\_request, 41  
  limine\_terminal\_request, 43  
  limine\_terminal\_response, 44  
revscroll  
  term\_context, 53  
rodata\_end\_addr  
  KernelUtils.c, 367  
rodata\_start\_addr  
  KernelUtils.c, 367  
rows  
  limine\_terminal, 42  
  term\_context, 53  
rrr  
  term\_context, 53  
rsp0  
  gdt.c, 125  
  gdt.h, 179  
save\_state

  term.c, 510  
  term\_context, 54  
saved\_cursor\_x  
  term\_context, 54  
saved\_cursor\_y  
  term\_context, 54  
saved\_state\_bold  
  term\_context, 54  
saved\_state\_current\_charset  
  term\_context, 54  
saved\_state\_current\_primary  
  term\_context, 54  
saved\_state\_reverse\_video  
  term\_context, 55  
ScopedLock, 46  
  ~ScopedLock, 46  
  ScopedLock, 46  
scroll  
  term\_context, 55  
scroll\_bottom\_margin  
  term\_context, 55  
scroll\_enabled  
  term\_context, 55  
scroll\_top\_margin  
  term\_context, 55  
second  
  time.c, 540  
  time.h, 335  
Segments  
  gdt.h, 178  
serial.c  
  serial\_print, 462  
  serial\_print\_line, 462  
serial.h  
  serial\_debug, 307  
  serial\_print, 307  
  serial\_print\_line, 308  
serial\_debug  
  gdt.c, 123  
  serial.h, 307  
serial\_print  
  serial.c, 462  
  serial.h, 307  
serial\_print\_line  
  serial.c, 462  
  serial.h, 308  
set\_cursor\_pos  
  term\_context, 55  
set\_fs\_base  
  cpuUtils.h, 155  
set\_gs\_base  
  cpuUtils.h, 155  
set\_kernel\_gs\_base  
  cpuUtils.h, 156  
set\_text\_bg  
  term\_context, 56  
set\_text\_bg\_bright  
  term\_context, 56

set\_text\_bg\_default  
     term\_context, 56  
 set\_text\_bg\_rgb  
     term\_context, 56  
 set\_text\_fg  
     term\_context, 56  
 set\_text\_fg\_bright  
     term\_context, 56  
 set\_text\_fg\_default  
     term\_context, 57  
 set\_text\_fg\_rgb  
     term\_context, 57  
 sgr  
     term.c, 511  
 shell.c  
     cmd\_parser, 464  
     vsh\_loop, 465  
     vsh\_readline, 465  
 shell.h  
     \_SHELL\_H, 309  
     VSH\_CMD\_BUFFER\_SIZE, 309  
     vsh\_loop, 310  
     vsh\_readline, 310  
     VSH\_VERSION, 309  
 SIZE  
     memUtils.h, 261  
 size  
     limine\_file, 20  
 sleep  
     time.c, 536  
     time.h, 332  
 smartlock, 254  
 smartlock\_acquire  
     lock.h, 256  
 SMARTLOCK\_INIT  
     lock.h, 254  
 smartlock\_release  
     lock.h, 256  
 smp\_req  
     KernelUtils.c, 368  
 snprintf  
     printf.h, 291  
 snprintf\_  
     printf.c, 442  
     printf.h, 297  
 spinlock\_acquire  
     lock.h, 256  
 SPINLOCK\_INIT  
     lock.h, 254  
 spinlock\_release  
     lock.h, 257  
 spinlock\_t  
     lock.h, 255  
 spinlock\_test\_and\_acq  
     lock.h, 257  
 split\_tag  
     liballoc.c, 382  
 sprintf  
     printf.h, 291  
     sprintf\_  
         printf.c, 443  
         printf.h, 298  
     src/alloc.cpp, 59, 63  
     src/cpudet-clean.c, 63, 68  
     src/cpuUtils.c, 72, 95  
     src/drivers/keyboard/keyboard.c, 99, 107  
     src/frameallocator.c, 109, 119  
     src/gdt.c, 122, 126  
     src/heap.c, 128, 132  
     src/idt.c, 135, 139  
     src/include(bitmap.h, 140, 142  
     src/include(cpuUtils.h, 142, 158  
     src/include/drivers/keyboard/keyboard.h, 161, 167  
     src/include/drivers/keyboard/keyboard\_map.h, 167, 169  
     src/include/gdt.h, 172, 179  
     src/include/global\_defs.h, 180, 182  
     src/include(heap.h, 183, 189  
     src/include(idt.h, 189, 195  
     src/include/interrupts.h, 196, 198  
     src/include/io.h, 199, 201  
     src/include/kernel.h, 202, 205  
     src/include/KernelUtils.h, 205, 211  
     src/include/lib/hashmap.h, 212, 216  
     src/include/lib/vector.h, 220, 226  
     src/include/liballoc.h, 228, 235  
     src/include/limine.h, 235, 248  
     src/include/lock.h, 252, 258  
     src/include/lock.hpp, 258, 259  
     src/include/memUtils.h, 260, 262  
     src/include/paging/frameallocator.h, 262, 270  
     src/include/paging/paging.h, 270, 278  
     src/include/paging/vmm.h, 279, 281  
     src/include/pic.h, 282, 289  
     src/include/printf.h, 290, 300  
     src/include/proc.h, 300, 302  
     src/include/registers.h, 303, 306  
     src/include/sched.h, 307  
     src/include/serial.h, 307, 308  
     src/include/shell.h, 309, 311  
     src/include/string.h, 312, 314  
     src/include/terminal/framebuffer.h, 315, 320  
     src/include/terminal/term.h, 322, 327  
     src/include/time.h, 329, 335  
     src/include/tss.h, 336, 337  
     src/include/vgafont.h, 338, 339  
     src/interrupts.c, 342, 345  
     src/io.c, 346, 348  
     src/kernel.c, 348, 357  
     src/KernelUtils.c, 360, 368  
     src/liballoc.c, 372, 383  
     src/lock.cpp, 389, 390  
     src/lock\_atm\_c.cpp, 390, 392  
     src/lock\_c.c, 393  
     src/memcmp.c, 393, 394  
     src/memcpy.c, 394, 395  
     src/memmove.c, 395, 396

src/memset.c, 396, 397  
src/memUtils.c, 398, 399  
src/paging.c, 399, 406  
src/pic.c, 408, 412  
src/printf.c, 413, 445  
src/proc.c, 457  
src/putchar.c, 457, 459  
src/register.c, 460, 461  
src/sched.c, 461  
src/serial.c, 461, 463  
src/shell.c, 463, 466  
src/strlen.c, 467, 469  
src/strtok.c, 469, 470  
src/terminal/framebuffer.c, 471, 491  
src/terminal/term.c, 500, 517  
src/time.c, 529, 541  
src/vmm.c, 544, 546  
stack\_size  
    limine\_stack\_size\_request, 41  
start\_interrupts  
    kernel.c, 355  
stop\_interrupts  
    kernel.c, 355  
string.h  
    memcmp, 312  
    memcpy, 312  
    memmove, 313  
    memset, 313  
    strlen, 313  
    strtok, 314  
strlen  
    string.h, 313  
    strlen.c, 468  
strlen.c  
    strlen, 468  
strtok  
    string.h, 314  
    strtok.c, 469  
strtok.c  
    strtok, 469  
swap\_palette  
    term\_context, 57  
symbol  
    KernelUtils.c, 361  
sys\_clock\_handler  
    time.c, 536  
    time.h, 333  
sysenter  
    cpuUtils.h, 158  
system\_readout  
    KernelUtils.c, 365  
    KernelUtils.h, 209  
system\_timer\_fractions  
    time.c, 540  
    time.h, 335  
system\_timer\_ms  
    time.c, 540  
    time.h, 335  
tab\_size  
    term\_context, 57  
task\_switch\_handler  
    time.c, 537  
task\_switch\_int  
    kernel.c, 355  
task\_timer\_count  
    time.c, 540  
term.c  
    CHARSET\_DEC\_SPECIAL, 501  
    CHARSET\_DEFAULT, 502  
    col256, 516  
    control\_sequence\_parse, 502  
    dec\_private\_parse, 503  
    dec\_special\_to\_cp437, 504  
    escape\_parse, 505  
    linux\_private\_parse, 507  
    mode\_toggle, 508  
    restore\_state, 509  
    save\_state, 510  
    sgr, 511  
    term\_context\_reinit, 512  
    term\_putchar, 513  
    term\_write, 515  
term.h  
    TERM\_CB\_BELL, 323  
    TERM\_CB\_DEC, 323  
    TERM\_CB\_KBD\_LEDS, 323  
    TERM\_CB\_LINUX, 323  
    TERM\_CB\_MODE, 323  
    TERM\_CB\_POS\_REPORT, 323  
    TERM\_CB\_PRIVATE\_ID, 324  
    TERM\_CB\_STATUS\_REPORT, 324  
    term\_context\_reinit, 324  
    TERM\_MAX\_ESC\_VALUES, 324  
    term\_write, 325  
term\_bg  
    kernel.c, 356  
TERM\_CB\_BELL  
    term.h, 323  
TERM\_CB\_DEC  
    term.h, 323  
TERM\_CB\_KBD\_LEDS  
    term.h, 323  
TERM\_CB\_LINUX  
    term.h, 323  
TERM\_CB\_MODE  
    term.h, 323  
TERM\_CB\_POS\_REPORT  
    term.h, 323  
TERM\_CB\_PRIVATE\_ID  
    term.h, 324  
TERM\_CB\_STATUS\_REPORT  
    term.h, 324  
term\_context, 47  
    autoflush, 48  
    bold, 48  
    callback, 49

charsets, 49  
 clear, 49  
 cols, 49  
 control\_sequence, 49  
 csi, 49  
 current\_charset, 50  
 current\_primary, 50  
 dec\_private, 50  
 deinit, 50  
 disable\_cursor, 50  
 discard\_next, 50  
 double\_buffer\_flush, 51  
 enable\_cursor, 51  
 esc\_values, 51  
 esc\_values\_i, 51  
 escape, 51  
 escape\_offset, 51  
 full\_refresh, 52  
 g\_select, 52  
 get\_cursor\_pos, 52  
 in\_bootloader, 52  
 insert\_mode, 52  
 kernel.c, 357  
 KernelUtils.h, 211  
 move\_character, 52  
 raw\_putchar, 53  
 restore\_state, 53  
 reverse\_video, 53  
 revscroll, 53  
 rows, 53  
 rrr, 53  
 save\_state, 54  
 saved\_cursor\_x, 54  
 saved\_cursor\_y, 54  
 saved\_state\_bold, 54  
 saved\_state\_current\_charset, 54  
 saved\_state\_current\_primary, 54  
 saved\_state\_reverse\_video, 55  
 scroll, 55  
 scroll\_bottom\_margin, 55  
 scroll\_enabled, 55  
 scroll\_top\_margin, 55  
 set\_cursor\_pos, 55  
 set\_text\_bg, 56  
 set\_text\_bg\_bright, 56  
 set\_text\_bg\_default, 56  
 set\_text\_bg\_rgb, 56  
 set\_text\_fg, 56  
 set\_text\_fg\_bright, 56  
 set\_text\_fg\_default, 57  
 set\_text\_fg\_rgb, 57  
 swap\_palette, 57  
 tab\_size, 57  
 term\_context\_reinit  
     term.c, 512  
     term.h, 324  
 term\_fg  
     kernel.c, 357

TERM\_MAX\_ESC\_VALUES  
     term.h, 324

term\_putchar  
     term.c, 513

term\_write  
     term.c, 515  
     term.h, 325

terminal\_count  
     limine\_terminal\_response, 44

test\_proc  
     vmm.c, 545

test\_table  
     kernel.c, 357

text\_end\_addr  
     KernelUtils.c, 368

text\_start\_addr  
     KernelUtils.c, 368

tftp\_ip  
     limine\_file, 20

tftp\_port  
     limine\_file, 21

this\_cpu  
     cpuUtils.h, 156

time.c

- century\_register, 538
- CHANNEL\_ZERO, 530
- CMOS\_ADDR, 530
- CMOS\_DATA, 530
- config\_PIT, 531
- count\_down, 538
- CURRENT\_YEAR, 530
- day, 538
- dayofweek, 531
- delta\_int, 531
- get\_RTC\_register, 532
- get\_update\_flag, 532
- hour, 539
- init\_PIT, 533
- minute, 539
- MODE\_CMD, 530
- month, 539
- month\_str, 539
- pit\_armed, 539
- print\_date, 534
- print\_sys\_time, 534
- read\_rtc, 535
- second, 540
- sleep, 536
- sys\_clock\_handler, 536
- system\_timer\_fractions, 540
- system\_timer\_ms, 540
- task\_switch\_handler, 537
- task\_timer\_count, 540
- timer\_irq, 538
- weekday, 540
- year, 540

time.h  
     \_TIME\_H, 329

day, 334  
hour, 334  
init\_PIT, 330  
minute, 334  
month, 334  
pit\_armed, 334  
print\_date, 330  
print\_sys\_time, 331  
read\_RTC, 332  
second, 335  
sleep, 332  
sys\_clock\_handler, 333  
system\_timer\_fractions, 335  
system\_timer\_ms, 335  
year, 335  
timer\_function  
    cpu\_local, 8  
timer\_irq  
    time.c, 538  
TranslateToHighHalfMemoryAddress  
    paging.h, 277  
TranslateToKernelMemoryAddress  
    paging.h, 277  
TranslateToKernelPhysicalMemoryAddress  
    paging.h, 278  
TranslateToPhysicalMemoryAddress  
    paging.h, 278  
TSS, 336  
tss  
    cpu\_local, 8  
    gdt.c, 125  
tss.h  
    \_TSS\_H, 337  
TSS\_Entry, 175  
TssStack  
    gdt.c, 125  
Unlock  
    AtomicLock, 6  
unused  
    limine\_file, 21  
    limine\_framebuffer, 24  
used\_ram  
    frameallocator.c, 117  
    frameallocator.h, 269  
usedMemory  
    frameallocator.c, 118  
vector.h  
    VECTOR\_ENSURE\_LENGTH, 221  
    VECTOR\_FIND, 222  
    VECTOR\_FOR\_EACH, 222  
    VECTOR\_INIT, 223  
    VECTOR\_INSERT, 223  
    VECTOR\_INVALID\_INDEX, 223  
    VECTOR\_ITEM, 224  
    VECTOR\_PUSH\_BACK, 224  
    VECTOR\_PUSH\_FRONT, 224  
    VECTOR\_REMOVE, 224  
    VECTOR\_REMOVE\_BY\_VALUE, 225  
    VECTOR\_TYPE, 225  
    VECTOR\_ENSURE\_LENGTH  
        vector.h, 221  
    VECTOR\_FIND  
        vector.h, 222  
    VECTOR\_FOR\_EACH  
        vector.h, 222  
    VECTOR\_INIT  
        vector.h, 223  
    VECTOR\_INSERT  
        vector.h, 223  
    VECTOR\_INVALID\_INDEX  
        vector.h, 223  
    VECTOR\_ITEM  
        vector.h, 224  
    VECTOR\_PUSH\_BACK  
        vector.h, 224  
    VECTOR\_PUSH\_FRONT  
        vector.h, 224  
    VECTOR\_REMOVE  
        vector.h, 224  
    VECTOR\_REMOVE\_BY\_VALUE  
        vector.h, 225  
    VECTOR\_TYPE  
        vector.h, 225  
vectors  
    idt.c, 138  
vendor\_str1  
    cpuUtils.c, 93  
vendor\_str2  
    cpuUtils.c, 94  
vendor\_str3  
    cpuUtils.c, 94  
vgafont  
    vgafont.h, 338  
vgafont.h  
    vgafont, 338  
VirtualAddressToOffsets  
    paging.c, 405  
vmm.c  
    proc\_page\_size, 545  
    test\_proc, 545  
    vmm\_page\_table, 545  
    VMM\_table\_clone, 544  
vmm.h  
    VMM\_table\_clone, 281  
vmm\_page\_table  
    vmm.c, 545  
VMM\_table\_clone  
    vmm.c, 544  
    vmm.h, 281  
vprintf  
    printf.h, 292  
vprintf\_  
    printf.c, 443  
    printf.h, 298  
VSH\_CMD\_BUFFER\_SIZE

shell.h, 309  
vsh\_loop  
    shell.c, 465  
    shell.h, 310  
vsh\_readline  
    shell.c, 465  
    shell.h, 310  
VSH\_VERSION  
    shell.h, 309  
vsnprintf  
    printf.h, 292  
vsnprintf\_  
    printf.c, 444  
    printf.h, 299  
  
weekday  
    time.c, 540  
White  
    kernel.c, 351  
width  
    limine\_framebuffer, 24  
writeCR0  
    registers.h, 305  
WriteCR3  
    paging.c, 405  
writeCR3  
    registers.h, 305  
writeCR4  
    registers.h, 305  
writeMSR  
    registers.c, 461  
    registers.h, 306  
wrmsr  
    cpuUtils.h, 156  
  
xrstor  
    cpuUtils.h, 157  
xsave  
    cpuUtils.h, 157  
  
year  
    time.c, 540  
    time.h, 335  
Yellow  
    kernel.c, 351